Table of Contents

Chapter 1 Ethics and Professional Conduct 2
Chapter 2 How to Succeed in the Computer Science Major 4
Chapter 3 Department Faculty Biography 7
Chapter 4 KU Systems 13
Chapter 5 Introduction to Unix 24
Chapter 6 Computer Science Curriculum 41
Chapter 7 Student Organizations Related to Computer Science 44
Appendix A 46
Computer Science Degree Checksheets 46
Appendix B Recommended Course Sequences 69
Appendix C 71
Tentative Course Offerings 71

Chapter 1 Ethics and Professional Conduct

Compliance with the Department Academic Integrity Policy and the ACM Code of Ethics (http://www.acm.org/about/code-of-ethics) is expected of all students in the computer science department.

Department Academic Integrity Policy[1]

Academic integrity is the pursuit of scholarly activity free from fraud and deception. Academic dishonesty includes, but is not limited to, cheating, plagiarism, fabrication of information or citations, facilitating acts of academic dishonesty by others, unauthorized possession of examinations, submitting work for another person or work previously used without informing the instructor or tampering with academic work of other students.

Maintaining the value of a degree from this institution requires a collective effort on the part of students and faculty to prevent the occurrence of acts that would cause harm to its reputation. Academic integrity requires a commitment from students not to engage in or tolerate acts of falsification, misrepresentation or deception.

The following defines ethics in the context of computer science.

Cheating/plagiarism: Since this discipline requires much out of class work, there needs to be clarification as to what constitutes cheating as opposed to what constitutes valid peer help. We recognize the benefit of peer help, within a certain context.

Acceptable peer help consists of the following:

- General discussion of the problem and the expectations of the professor
- Discussion of algorithms
- Using pseudo code to help another understand
- Looking at another person's already written code to help them find a problem.

Cheating/plagiarism consists of any kind of sharing of any piece of code that the person is supposed to be doing on his or her own. This would include providing code to another orally, with a hard copy, via email or in any other manner, including sitting side by side entering the same code. Cheating/plagiarism is also committed if a student copies code from a web page, help forum, or other source on the Internet. Sharing of code can be initiated by someone trying to help the person or having the person steal the code, such as taking a printed copy from the trash or copying it from an accessible location.

This list is not intended to be exhaustive. If you aren't sure what you are doing is okay, **don't do it**. Your professor is the final arbiter of what constitutes academic dishonesty in her/his class.

[1] http://cs.kutztown.edu/pdfs/AcademicIntegrityPolicy.pdf

The kind of cheating that takes place in the field of computer science generally involves a giver and a receiver of code or solutions. Both are considered to have participated in the cheating when it occurs. Both will receive the consequences set up by the university and the professor.

Activity free from fraud and deceptions: The receiver of inappropriate help is committing fraud and/or deception by indicating to the professor that work is hers/his when it is not.

Facilitating acts of academic dishonesty: The provider of inappropriate help is facilitating academic dishonesty by another by allowing a student who may not have been able to do the work to commit fraud as described above.

Failure to uphold the described standard of academic integrity is a serious offense. If a student has cheated, the university Academic Honesty Policy states the faculty member should complete an Academic Dishonesty Report Form. An informal resolution process is initiated where any in course sanction may be imposed up to and including failure of the course. A formal process through the Office of Judicial Services exists for flagrant or multiple violations where a potential outcome is suspension or dismissal from the university.

The Computer Science Department is committed to preventing academic dishonesty and punishing perpetrators appropriately.

Chapter 2 How to Succeed in the Computer Science Major

How to Succeed in Computer Science

Cultivate appreciation for your work

It is important to cultivate appreciation for your work, both the process of doing it and the results that you achieve. Designing, assembling, testing and debugging individual programs or information processing systems takes a lot of work. For you to perform the necessary work without shortcuts or procrastination that harm its quality, you have to like doing it. You don't have to like it all – there is always some "grunt work" that just needs to get done – but you basically have to like what you are doing in order to succeed in this or any major.

Presumably you like working with computers, or you wouldn't be reading this. It is also likely that you have used software tools that typical home PC users do not typically use. You certainly know how to use tools for interacting with people and information on the Internet and the World Wide Web. If you can successfully extend that appreciation for these hardware and software systems into going "under the hood," finding out how they work, and ultimately extending how they work in new and useful ways, you will succeed.

There is a book called *Zen Mind, Beginner's Mind*. The essence of this book's message is that you "have it" at the beginning, and though you may "lose it" in going along, you can rediscover that you "have it," because in fact you never really lost it. ☺ You can certainly think of activities that initially attracted you, but over time lost their attraction, perhaps because you lost the feeling about those things that you had at the beginning. That feeling was part of Beginner's Mind, perhaps before you became jaded, because some aspect of the activity seemed to be too much work, or felt annoying in some way. You have the Beginner's Mind right now, and a big part of cultivating appreciation for your work comes down to this: Don't discard your Beginner's Mind! One way to avoid this is by not digging pits into which you then fall. Falling into these pits ruins your appreciation for your work. Avoid the pitfalls discussed in the following sections.

Start early

Procrastination is one of the biggest killers of grades in this major. One cause of procrastination may be that you do not like some aspect of the work, that you have lost your Beginner's Mind. Another, common cause is thinking that you don't need to start early because you have plenty of time. You *may* have plenty of time for the project that your professor has just given you, but the problem is that *you probably don't know that*. Experienced software engineers and IT professionals who are otherwise very good at their work, are often only passably good at estimating how long it will take them to finish a project. Experience helps improve this skill, but when projects contain unknown aspects, such as problems with tools and code libraries or unknown customer requirements, then it is hard to predict how long they will take to complete. Unlike professionals in this field, you do not have years of experience doing the things that you are about to do. That is why you are here. That means that problems and unknowns that will cost project time are likely to crop up for some projects, and unfortunately, you do not know in which projects they will appear.

Scheduling work time comes down to some simple math. Suppose a professor gives you a project assignment that will take you **H** hours to complete. The trouble is, you don't know the value for **H**, and so you make one up. Let's call it $\mathbf{H_{guess}}$.If it is really going to take you **H** hours to complete the project, then it doesn't make a difference whether you start it shortly after it is assigned, or a day or two before the project is due. It will take **H** hours. There is no more work entailed in starting early than in starting late. Suppose, though, that you start the project the day before it is due, and it turns out that **H** is bigger than the number of hours you have available to work on the project that day. Perhaps for your average project, $\mathbf{H_{guess}}$ is a good estimate of **H**, but for this project, $\mathbf{H_{guess}} < \mathbf{H}$, maybe a lot less. Congratulations! You have just dug a pit and fallen into it. Now, your project will be late or buggy or both, and your grade will suffer. Perhaps subsequent projects will build on this project. They depend on it working correctly. Possibly worst of all, you have just managed to turn your work into an unpleasant experience. A failed project conditions you to dislike your work, increasing the likelihood that you will feel like procrastinating on the next project when it arrives.

We are all busy, and we don't want to spend all of our time working. This work is important, though. It is worth learning how to do it correctly. You don't need to obsess or worry or spend all of your time working. You do need to cultivate good work habits and a critical work habit is to start early. Remember, **H** hours are **H** hours no matter when they start, but if they start early, then the occasionally really big **H** won't help to create a pit for you to fall into.

Come to class, on time, prepared, and pay attention while you are here.

Notice that it doesn't say, "Go to class," it says, "Come to class." Class isn't over there, somewhere, maybe somewhere you'd rather avoid. Class is here. Here is where you learn something. You are in the middle of your own learning. If you need or want to learn something, and you aren't in the middle of learning it, then you are in the middle of nowhere.

Studies show that showing up improves acquisition and retention of knowledge and skills. Showing up on time has a strong correlation with success. Students or workers who show up late are simply procrastinating about one more aspect of their work. They are here in body, but not in mind. That statement goes double for throwing away your time in the classroom by not paying attention. If you are browsing the Web in a lab class, or nodding off, or reading something else, or *not actively engaging the classroom activity*, then you are not entirely here. "Actively engaging the classroom activity," means actively listening, asking questions, sometimes challenging assertions made by your professor, taking notes, and otherwise going through the motions. *Going through the motions pays off*! It activates parts of your nervous system and other systems that are involved in learning. If all of our learning depended on the doings of our conscious minds, we'd probably still be waiting for someone to invent the wheel, let alone the computer. Maybe the inventors of the wheel saw a rounded stone or a log rolling down a hill. They had to be on the scene – the hillsides were their classrooms – and they had to be paying attention.

Sure, class is boring some times. Making that fact the basis for your decision to blow off class digs another pit into which you can fall. Boring or not, maybe today's class is the one in which you will become enlightened. Typically, every student shows up on the first day of class.

They all have recovered the Beginner's Mind! Show up, and if it seems boring, make it less so by actively engaging the professor, and other students in a collaborative class session, with questions and answers and observations. Maintaining this habit will help you succeed in your work and improve your enjoyment.

It is important to be prepared for class. Be sure to do all reading and assignments before class. If you don't understand something while reading the text, pursue that topic further, using other resources, and see if you can understand it better before asking the professor in class. If the professor makes PowerPoint or notes available for the lectures, utilize them before class. It will help you prepare for learning. Computer science is a major that requires constant learning, often on your own. Build that habit now and it will be one that will stay with you.

Talk with your professors.

Not just in class, but also outside, talk with your professors. If you have questions or problems with projects or don't understand something, visit your professor during office hours, or schedule a time to meet, or at least send email. It's fine to do these things with your peers – although I doubt that your peers have office hours – but don't miss the opportunity to talk with your professor, one on one. It's the best student-to-teacher ratio that you'll ever get. Especially when you are stuck on some part of a project, discussing it with your professor is probably the best way to avoid another pitfall. Please note, though, that working through project problems with your professor means starting early, so you can schedule a meeting if you get stuck. If you start at the last minute, and send last minute emails that don't receive last minute replies, then you are digging another pit. Start early, and if you get stuck, go see your professor.

Talking to your professor and peers is an extremely important aspect of learning in the field of computer science. At the same time, the effort you apply to solving the problem yourself will give you insight into solving the problem. It may be the case that you get stuck at some point. The question is how long do you try to get unstuck before you speak with your professors and/or peers. There is no rule of time, but if you have done everything you know to get unstuck, then it would be time to speak with others about the problem you cannot solve.

There are also Graduate Assistants available in the Computer Science Department to help you with problems in your courses. Kutztown University offers free tutoring services for students. If you were successful in a course, sign up to be a tutor for that course in a later semester. Teaching a topic to a peer is the best way to completely understand it.

Finally, invent interesting computer projects for yourself and your friends.

Make these machines your own! Maybe this means collaborating in club activities such as the Kutztown Technology Association or the Mobile Apps Development Club. Maybe it means inventing computer games or composing computer music or enhancing a web site or helping someone else with their computing needs, perhaps by fixing a friend's wireless network problems, or by becoming a tutor. Not everything gets done in regular courses and classes. Some of the best computing activities may be your own inventions. You may be able to earn independent study credits for some of these activities. If not, you can still improve your appreciation of the discipline, and beef up your resume, and maintain your Beginner's Mind by making these machines your own, in and out of the classroom.

Chapter 3 Department Faculty Biography

Prof. Linda L. Day holds a Bachelor of Science degree in Mathematics from Western Illinois University and a M.S. degree in Computer Science from Lehigh University. Before coming to Kutztown University in 1985 she worked two years for AT&T Information Systems in Lincroft, NY. Professor Day has served as chair of the Computer Science Department since July of 2007. She has taught courses in computer applications, computer science I and II, data structures, object oriented programming and file organization. Her interests include object oriented design and analysis and human computer interface design.

Dr. Lisa M. Frye received her Ph.D. in Computer Science from Lehigh University in January 2012. She is a 1993 and 1990 graduate of Kutztown University with a Master of Science degree in Computer Science and bachelor's degrees in Secondary Education/Mathematics and Computer Science. Dr. Frye has a background full of industrial experience prior to joining the Computer Science Department faculty. Her professional career includes experience as an industrial software engineer, systems engineer, systems and network administrator, and support analyst for local industry and educational institutions. Her research interests include networking, particularly network management and network security, as well as Computer Science education. Dr. Frye joined the Computer Science Department at Kutztown University in 2001. In the classroom, she introduces students to the complex world of computer science, teaching both introductory courses and advanced studies in programming, networking, security, and web development.

Dr. Tauqeer Hussain holds a Bachelor of Science degree in Chemical Engineering from University of the Punjab. He received his M.S. degree in Operations Research from Wayne State University with CGPA 4.0/4.0 and Ph.D. degree in Computer Science from Lahore University of Management Sciences. Before coming to Kutztown University in 2009, Dr. Hussain worked as Systems Analyst, Software Engineer and Project Manager for various companies for over 15 years. He taught at many universities for over 20 years, and also served at various administrative positions in academia. His courses at Kutztown University focus on Information Technology, Programming, Database Design, Computer Graphics, and Project Management. His areas of interest are Software Engineering, Database Systems, Operating Systems, Operations Research, Information Systems, and Project Management.

Dr. Randy Kaplan graduated from Drexel University with a B.S. in Mathematics and Temple University with an M.A. in Computer Science and a Ph.D. in Computer Science. While working on his Ph.D. he specialized in the subfield of Artificial Intelligence. His research focused on how to extract knowledge from text in an effort to automatically build a knowledgebase from text. Dr. Kaplan went on to apply this work at Educational Testing Service where he created the first prototype of a system that would grade student essays. Today he holds a patent for this system. For most of Dr. Kaplan's career he has acted as a teacher to those around him. At Kutztown University he is able to share his unique perspective and experience with Computer Science students.

Dr. Dale E. Parson holds a Bachelor of Science degree in computer science from Albright College and M.S. and Ph.D. degrees in Computer Science from Lehigh University. Before coming to Kutztown University in 2008 Dr. Parson worked for Bell Labs and related companies for 28 years. Dr. Parson's software engineering work included research and development in design automation for integrated circuit design, software development tools for digital signal processors, architecture and applications for network processors, and software architecture for multimedia server devices. His courses at Kutztown focus on computer graphics, data structures, programming languages, compilers and programming for multiprocessors. His primary research area is computer music instrumentation and improvisation.

Dr. Oskars J. Rieksts holds a Bachelor's degree in mathematics, Master's degrees in philosophy of science and in mathematical sciences, and the Ph.D. degree in computer science from Eastern University, Indiana University, Virginia Commonwealth University and the University of Delaware, respectively. He was a Fulbright Scholar at the University of Tubingen and studied at Biblical Seminary. He taught at Piedmont Christian School and Virginia Commonwealth University before coming to Kutztown University in 1984. He has taught introduction to computer science I & II, discrete math I & II, information technology fundamentals, Visual Basic and Advance Visual Basic, operating systems and advanced operating systems, theory of computation, networks, information security, applied cryptography, artificial intelligence I & II, intelligent robotics, and seminar in computer science. His research interests are in design theory as applied to computer systems, intelligent robotics and cryptography. All three of his sons graduated from Kutztown.

Dr. Greg Schaper earned a B.S. in Computer Science from Arkansas State University in 1983 and a Ph.D. in Computer Science from the University of Central Florida in 1989. His academic career in Computer Science includes positions as Graduate Teaching Assistant at Arkansas State University, Graduate Teaching Assistant, Adjunct, and Visiting Instructor at the University of Central Florida, Associate Professor at East Tennessee State University, Temporary Assistant Professor at the University of Hawaii, Associate Professor at Hawaii Pacific University, and is currently an Associate Professor at Kutztown University. He has published or presented research in a variety of areas including VLSI Chip Design, Graph Theory, Computational Complexity, Parallel Processing, Neural Networks, Real-Time Battlefield Simulation, Robotics, and Computer Science Education. While his research interests are diverse, they are currently focused in the areas of Computer Science Education and Curriculum Development.

Dr. Yong-Sang Shim holds a Bachelor of Science degree in Mathematics from Yeungnam University in Korea. Dr. Shim holds M.S. and Ph.D. degrees in Computer Science from South Dakota State University and the University of Wyoming respectively. His courses at Kutztown University focus on information technology systems, information security, programming languages, database, algorithms, and discrete mathematics. His primary research area is computer security, system modeling, and computer education.

Dr. Daniel Spiegel, Professor of Computer Science and coordinator of the Master's program, received his Ph.D. from Wright State University in 2002, an M.S. in Mathematics with a concentration in Computer Science, and a B.S. in Mathematics from The Ohio University in 1987 and 1982, respectively. Prior to being hired at Kutztown University, he was on the faculty of the Department of Computer Science & Engineering at Wright State. He was an instructor in the Computer Science Department at Ohio University before that and also was an instructor for Hocking Technical College. His research interests include evolutionary techniques, including genetic computing, operations research, issues in computer science education, and biological simulation. He has also directed theses in diverse areas including lossless streaming and data mining.

Dr. Joo Tan holds a Bachelor's degree in Computer Science from the University of New Mexico and the M.Sc. and Ph.D. degrees from Kansas State University. He had a previous teaching stint at Mansfield University. Prior to academia, Dr. Tan worked at various companies such as BELLCORE and Lucent Technologies for 6 years. His courses at Kutztown University focus on Introductory Computer Applications, Web site design, and Software Engineering. His research interest is in the latest technology of web design and programming as well as collaborative team learning in Software Engineering.

Dr. Yong Zhang holds a Bachelor's degree in chemistry from the University of Science and Technology of China and M.E. and Ph.D. degrees from the University of South Carolina. He taught at Eastern Mennonite University for three years prior to joining Kutztown University in Fall 2008. He regularly teaches courses in C++ programming, discrete mathematics, data structure, algorithm analysis, and applied cryptography. His research interest is mainly in theoretical aspects of computer science, including algorithm design, computational complexity, and quantum computation.

Chapter 4 KU Systems

Introduction

The Computer Science Department has many different systems available for use by our students. There are labs available for general use by computer science students, as well as some specialized labs for specific courses. The computer science general lab is in Old Main 248A and contains primarily PCs with a few Macs. There are also many labs around campus that can be used to access the necessary systems and software. The location of these labs and the hardware and software available in these labs can be found on the Office of Information Technology's web page (http://it.office.kutztown.edu/).

UNIX Systems

The Computer Science Department utilizes UNIX for many of the courses. There are a several UNIX systems available for use by our students. The primary UNIX system, at this time, is ***bill***. This system is also known as ***login*** *or* ***acad***. This system is a Sun Fire V245 server running the Solaris operating system. It has one 1.5 Ghz processor and 8 GB of memory.
Two other UNIX systems that are widely used in specific courses are ***yin*** and ***yang***. Both of these systems are also Sun systems running the Solaris operating system. Each one has a 1 Ghz processor. ***yin*** has 4 GB of memory and ***yang*** has 2 GB of memory. There are three UNIX systems granted by Sun Microsystems to support multiprocessing research and courses. These systems are ***harry***, ***Hermione***, and ***ron***. Table 4.1 shows the specifications of each of these servers.

name	Architecture	Cores	Threads / core	Clock Speed	Memory	Operating System
Harry	UltraSparc T2, T5120 server	8	8	1.2 Ghz	16 GB	Solaris
Hermione	AMD Opteron 885, x64, X4600 server	8	2	2.7 Ghz	32 GB	Linux
Ron	UltraSparc T1, T1000 server	8	4	1 Ghz	8 GB	Solaris

Table 4.1: Specifications of the Specialized Multiprocessor Servers

To access the UNIX systems, which are located in various buildings on campus, terminal emulation software must be used. This software must support the ssh protocol. The most common terminal emulation software used at KU is called PuTTY. This is simply software that allows a user to connect to a UNIX system.

Fig. 4.1 is a diagram of the KU UNIX systems. Each UNIX system connects to the KU network via a switch. Each computer on campus, desktop or laptop, also connects to the KU network via a switch. This allows the computers on campus to connect to the UNIX systems allowing students to login to these systems. When you connect to the Internet from your computer off-campus, this allows you access to certain systems on the KU Network, including

the UNIX systems. It does not matter where you connect from, on- or off-campus, you will connect to the same UNIX system(s) in the same manner.

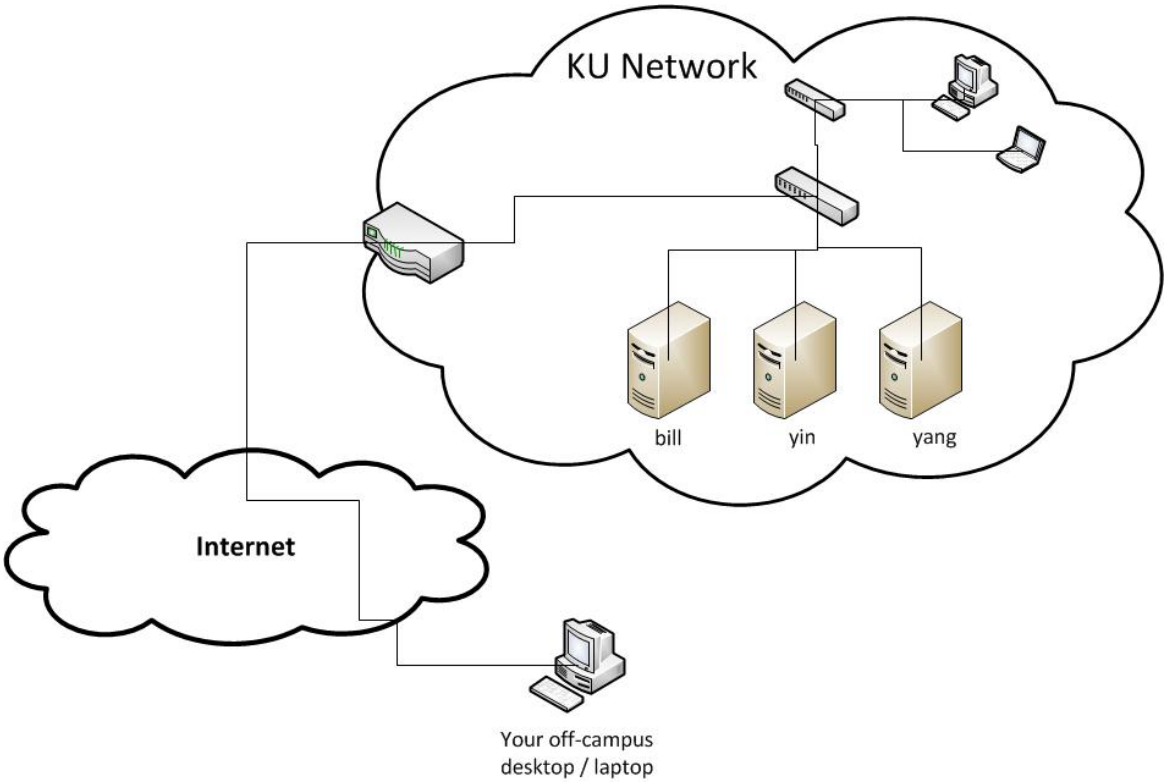


Figure 4.1: KU Systems Diagram

Your Account

All KU computer science students are provided with an account to access the Windows and UNIX systems on campus. The Windows account, which provides you access to the lab PCs, is activated when you become a KU student. The UNIX account is activated when you take your first computer science course at KU. Both accounts use the same username or login, which is your KU "Network ID". This account is an Active Directory account. Your Network ID consists of the first initial of your first name, the first four characters of your last name, and a random three-digit number. For example, Jane Smith's Network ID would be jsmit123.

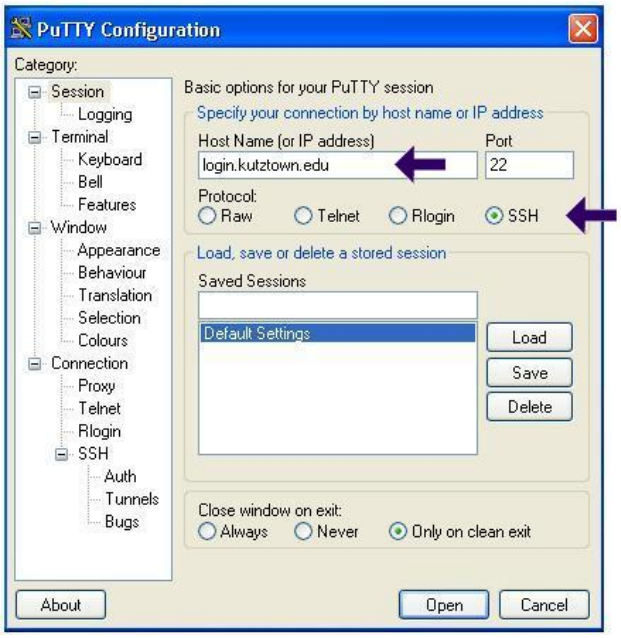
To change the password on this account, visit "Student Account Management" web page at https://sam.app.kutztown.edu/. If you forgot your password, and you have populated your security questions, you can reset your password at https://sam.app.kutztown.edu/forgot/index.aspx.

If you did not populate your security questions, you will have to go to the Office of Information Technology's Helpdesk located in the Administration Building.

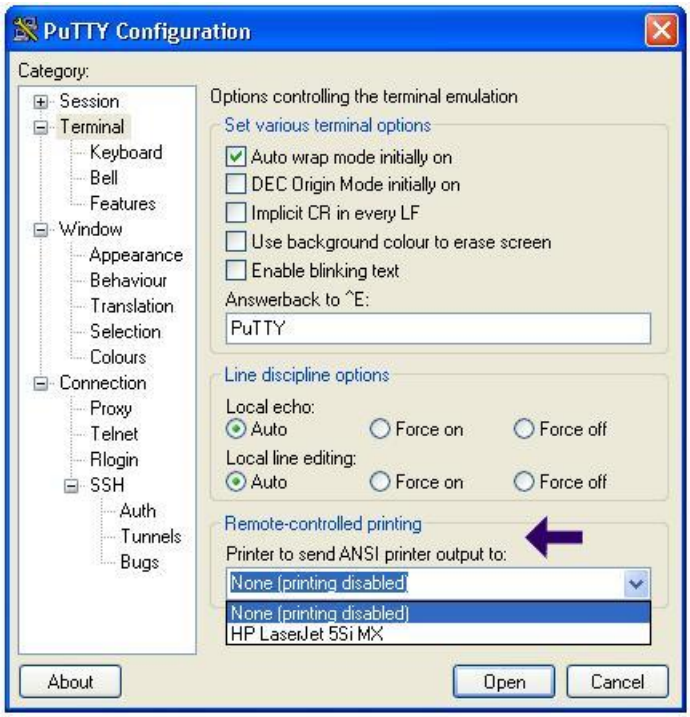
Accessing the UNIX Systems

The UNIX systems can be accessed from on- and off-campus using terminal emulation software. The most common software for Windows machines is PuTTY. Information on downloading and using PuTTY is provided here. The most updated information can be found at http://www.kutztown.edu/infotech/divisions/servers/unix/putty.shtml.

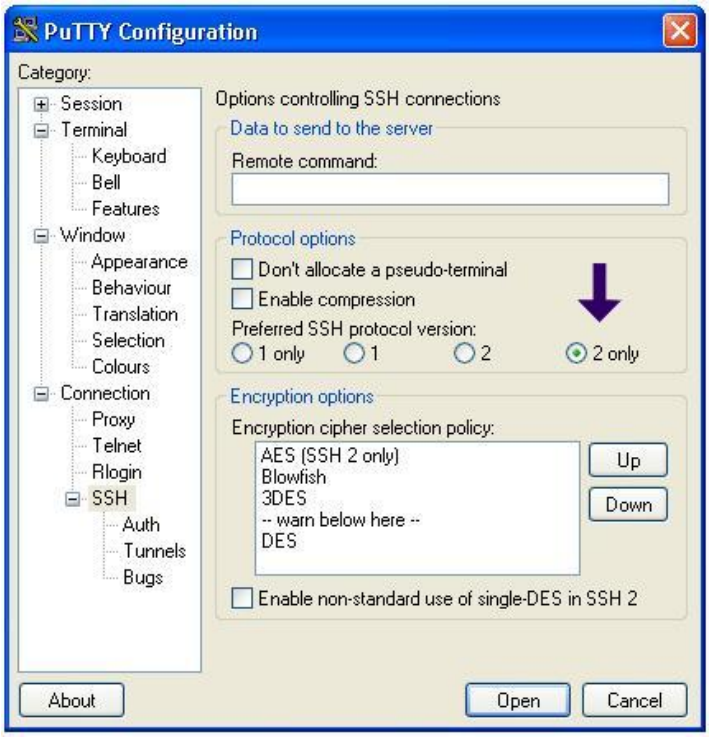
To use PuTTY with Windows you must first download it from http://download.kutztown.edu/. Use your Network ID to login to this web page. You may find additional information on PuTTY in the user manual located at http://the.earth.li/~sgtatham/putty/0.53b/htmldoc/Contents.html. When you download PuTTY, it is one file; there is nothing to be installed. Simply save the file to a location and then double-click on that file to run PuTTY. Follow these instructions to configure PuTTY for use with KU's primary UNIX system (bill).



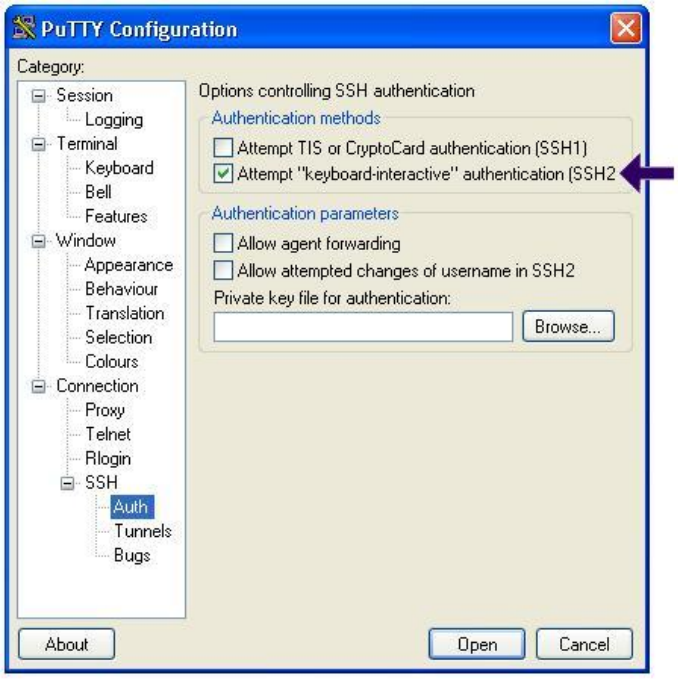
Enter login.kutztown.edu as the Host Name. Under Protocol, select SSH.



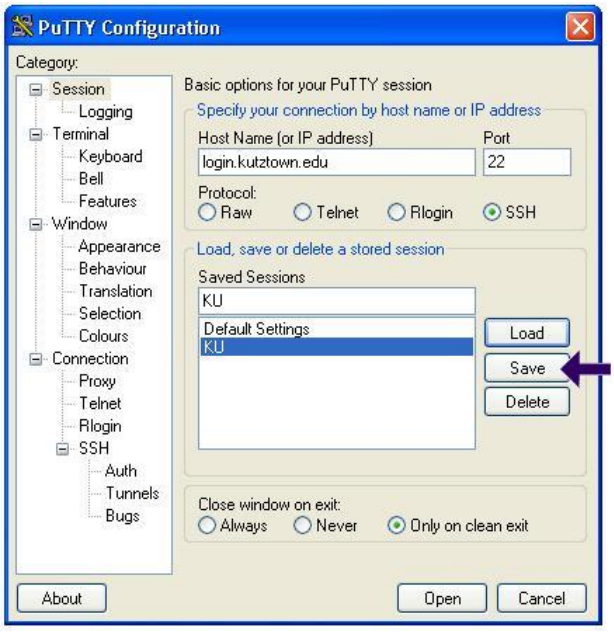
Click Terminal. Under Remote-controlled printing, select the name of your local printer.



Click Connection. Click SSH. Under Protocol options, select 2 only.



Click Auth and check “Attempt keyboard-interactive authentication(SSH2)”.



Click Session. Type KU in the Saved Sessions box. Click Save. Double-Click KU to launch.

PuTTY Security Alert
The server's host key is not cached in the registry. You have no guarantee that the server is the computer you think it is.
The server's key fingerprint is:
1024 50:60:aa:30:9b:89:3f:f2:9a:ce:f5:e8:99:ba:45:54
If you trust this host, hit Yes to add the key to PuTTY's cache and carry on connecting.
If you want to carry on connecting just once, without adding the key to the cache, hit No.
If you do not trust this host, hit Cancel to abandon the connection.
Yes
No
Cancel

The above alert may appear the first time you connect to login. Simply click Yes.

Accessing the UNIX systems using a Mac

To access the UNIX systems using a Mac, there is no additional software required. The Mac operating system has software incorporated in it that uses ssh. First you must launch the Terminal application. In the command-line prompt, enter *ssh -2 networkID@login.kutztown.edu* where *networkID* is your KU Network ID. For example, Jane Smith (jsmit123) will enter ssh -2 jsmit123@login.kutztown.edu. This will connect you to the primary UNIX system (***bill***). To connect to a different UNIX system, replace ***login*** with the name of the UNIX system you want to connect.

If you encounter problems with being able to use an editor due to an error with your terminal type, you must configure your terminal. To do this, login to your UNIX account and edit the file called ***.login*** located in your home directory. Add this line to that file and then save the file:

setenv TERM xterm

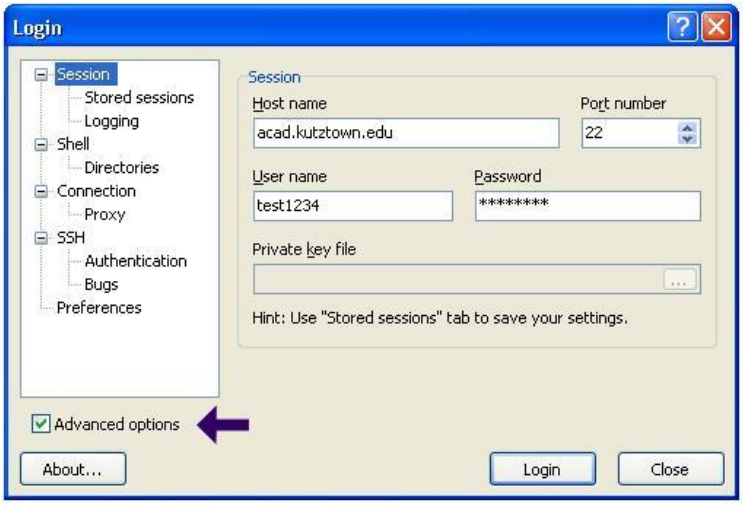
If you use a shell other than the default csh, such as bash or sh, then you must edit the appropriate login file. For bash, this file is .bash_profile and for sh the file is .profile. The line to add to the appropriate login file for all shells except csh is:

export TERM=xterm

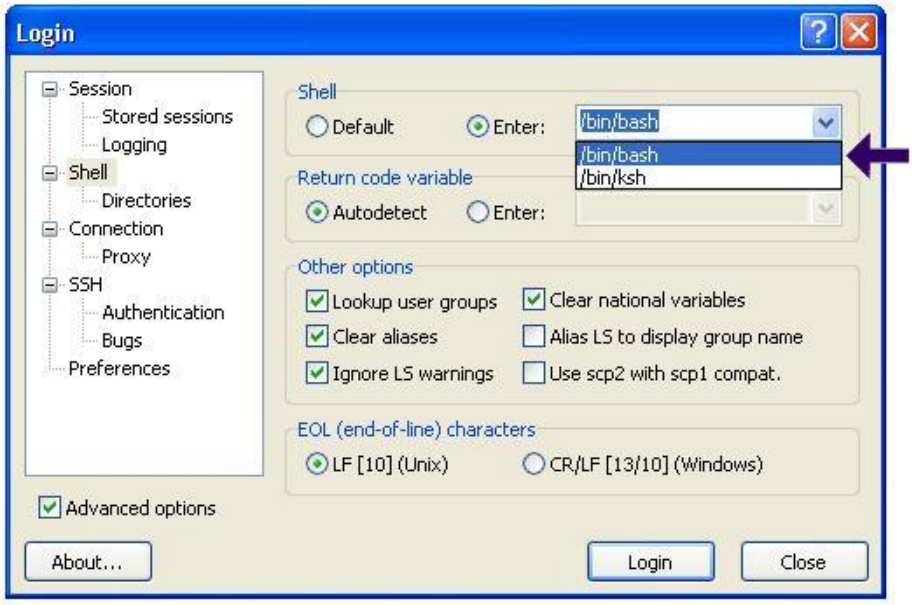
You can use dtterm instead of xterm in the above commands (the *setenv* and *export* commands) to get programming-language specific highlighting in some text editors.

File Transfers

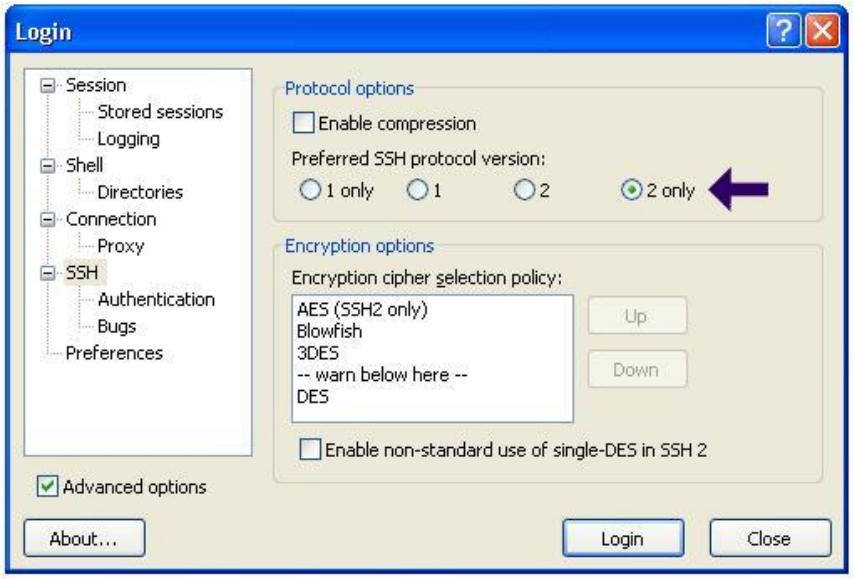
To transfer files to or from a UNIX system, you must use a file transfer software that supports the ssh protocol. The file transfer software used at KU is WinSCP. You can download this software from http://download.kutztown.edu/. To use WinSCP, follow these instructions, available at http://www.kutztown.edu/infotech/divisions/servers/unix/winscp2.shtml.



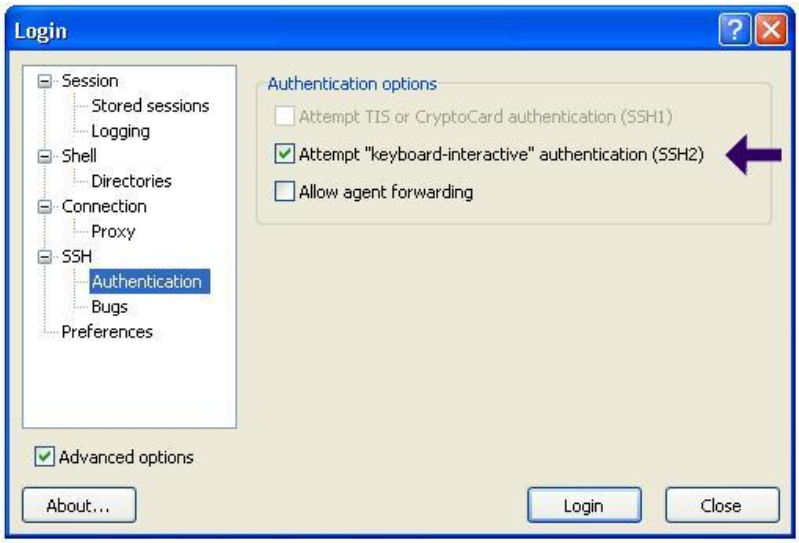
Check “Advanced Options”, if not already checked. Under the Session, enter ***login.kutztown.edu*** in the “Host name” field and make sure the “Port number” is 22. In the “User name” field enter your KU Network ID and then your Password.



Click on Shell: Select ‘Enter’ under Shell and in the drop down box, select ‘/bin/bash’.



Click on SSH and select '2 only'.



Under SSH, click on Authentication. Make sure "Attempt keyboard-interactive authentication (SSH2)" is checked. Click Login to login to this FTP program. These settings will be saved for the next time you run WinSCP.

File Transfers using a Mac

To transfer files to or from the UNIX systems using a Mac, there is no additional software required. The Mac operating system has FTP software incorporated in it. First you must launch the Terminal application. In the command-line prompt, enter *ftp –A login.kutztown.edu*.

This will then present you with a prompt to enter your username and then a prompt to enter your password.

You may also want to use other FTP software. A recommended FTP software is Filezilla, available at http://filezilla-project.org/.

Printing

You can print to any computer on campus from the UNIX system. To print from a lab computer, type the command *printers* at the UNIX prompt. This will list the name for each lab printer and also give you the commands to use to print. There are two commands that you can use to print: *lp* and *enscript.* The preferred command to use is *enscript* because it will wrap any line that is too long. The command to print to a printer on campus is

enscript -d printer_name filename

For example, to print the file program1.cpp to the printer in the Computer Science lab in Old Main 248, the command would be

enscript -d CSD program1.cpp

If you have configured your printer in your terminal emulation program, you can print from home as well. The command to print to the printer configured on your PC is *lph*. The syntax is

lph filename

So, to print the same file as above, you would enter the command

lph program1.cpp

Turning in Programs Electronically

Many computer science instructors require you to submit your programs electronically using a program called *turnin*. *Turnin* takes the files specified in the command and places them in a subdirectory belonging to your course instructor that is reserved for your submissions. This will allow submission and grading of programs to occur as efficiently as possible.

Setting up turnin

To set up the turnin utility for your use, you will run a perl script that alters your .login file and creates or updates your .alias file. This script is to be run once only each term. To run this script, issue the following command from your root directory on the UNIX system (<section> is optional; only use if instructed by your professor):

/export/home/public/<prof>/turnin.pl <course> <section>

In this command, <prof> is your instructor's Email address, all lower case, without the "@kutztown.edu" and <course> is the course number. If you are in CSC 135, this argument would be 135 (no CSC). The <section> is your section number. This is necessary in order that your submission be placed in the area for your particular section of the course. The section number is optional, at the discretion of your professor, so only specify it if instructed to do so by your instructor.

For example, If you are in CSC 135 section 045 with Dr. Spiegel, you would set up turnin using the command:

/export/home/public/spiegel/turnin.pl 135

If Dr. Spiegel instructs you to use the section number, then you would use the command:

/export/home/public/spiegel/turnin.pl 135 045

If you use the bash shell, you must run the turnin.bash.sh script, which will make some additional modifications to your login file. Use this command:

/export/home/public/<prof>/turnin.bash.sh

Invoking turnin

For each programming project, you will submit all requested files using the turnin command, as follows:

turnin<class #> <file1> <file2> … <filen>

where <class #> is the CSC course number for this course (just the number, no CSC) and <filei> is the name of a file in the present directory or the name of a file that includes a full or relative pathname. There is no limit to the number of files that may be turned in via the turnin command. The names of the files are separated by whitespace (at least one blank); do not use any other character to delimit file names. Wildcards are permitted, but you can't submit an entire directory by the directory's name.

For example, if a student in CSC 135 intends to turn in the files project1.cpp and ReadMe.txt, the command would be:

turnin135 project1.cpp ReadMe.txt

Note that there is no blank between the turnin and the 135.

There are some important notes to remember when setting up and invoking turnin. The perl script turnin.pl must be run from your login, or root directory (the directory you start in when you log in to the UNIX system). If you run it from any other directory, you won't be able to use turnin. Remember, it is run only once per semester. You are responsible to set up turnin BEFORE the first project is due. Not setting up turnin is NOT a legitimate excuse to turn in work

late. After setting up turnin, either log out and log back in, or execute source turnin before attempting to use the utility. The section number used in setting up turnin must be a 3-digit number, even if the first digit is 0. You may turn in a file any number of times. Each submission will overwrite the previous version submitted. After each execution of turnin, you will receive an e-mail containing a listing of the files you have turned in (now or previously) for this project. If you don't receive this mail, it is possible that the turnin failed. Run turnin again. It is a good idea to save these Emails. Please report any difficulty with setting up or using turnin to the course instructor as quickly as possible.

Microsoft DreamSpark Program (formally known as MSDNAA)

The Microsoft DreamSpark Program (formally known as Microsoft Developer Network Academic Alliance, MSDNAA) is a program sponsored by Microsoft that allows academic institutions to provide access to Microsoft software in labs, classrooms, faculty, and student PCs. The Computer Science Department is a member of DreamSpark. This allows our faculty and students access to the latest Microsoft software for use in classes and research. Microsoft allows students free access to most of its software, including Operating Systems and development tools.

To obtain a DreamSpark account, send an E-mail to msdnaa@kutztown.edu. The subject should be “Account Request”. In the body of the message indicate your major and the computer science courses you are currently taking along with the course instructor.
KU’s DreamSpark site can be accessed from the Computer Science Department’s web page under the “Resources” link.

Chapter 5 Introduction to Unix

Introduction to Editors

An editor is a computer program with a special purpose. When you are writing programs an editor is used to create the file that contains your program text. Your program text is contained in a file type called a text file. A text file is a simple file that contains only the text that you enter. This is different from a file produce by a word processing program like Microsoft Word. Microsoft Word produces a file called a "doc" file. This type of file contains both the text that you enter and also the format of the text so that it can be displayed properly and printed properly. There is no formatting information stored in a text file.

There are two kinds of text editors, a character editor and a line editor. As the names imply a character editor is used to create and edit a file on a character-by-character basis. The commands used in a character editor manipulate the characters in a text file. A line editor's commands manipulate the text in a file on a line-by-line basis. Note that a line editor can also manipulate characters. The difference is that in a line editor there are commands that manipulate lines and blocks of lines of text. Of the following VI is a character editors, EMACS and PICO are line editors. All three of these editors are usually standard programs on UNIX/Linux.

EMACS

EMACS is a full-featured editor that has evolved over the years into a very sophisticated editor. The EMACS editor is so sophisticated that a complete programming language is built into EMACS. The EMACS editor can be customized for use with any programming language. Many customizations already exist.

The EMACS editor makes use of key-chords as does the PICO editor. A key-chord consists of two or more keys pressed at the same time.

To start up the EMACS editor use the UNIX command "**EMACS**" (all lower case). This command is entered at the UNIX command prompt

EMACS filename (press enter)

where filename is the name of the file to be created (if it doesn't exist) or edited (if it does exist). When you enter this command the screen displayed is shown in Figure 5.1

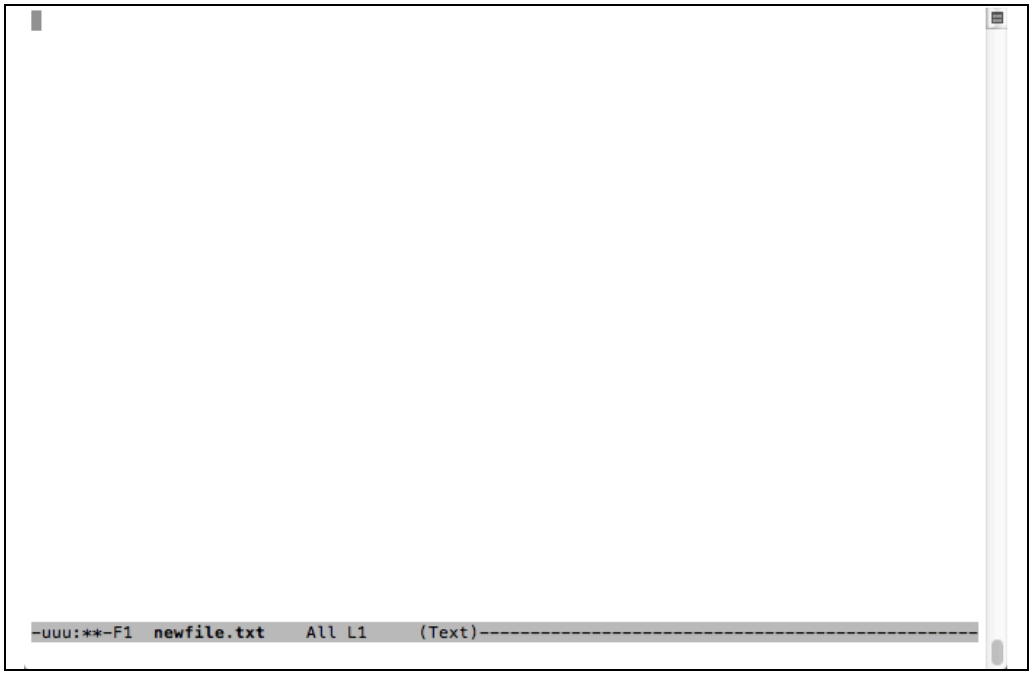


Figure 5.1 The EMACS screen at startup

The last line of the display is called the mode line. The mode line will always be near the bottom of the display window. It contains information about the file being edited and the state of the editor. The two asterisks near the beginning of the line indicate that the buffer that contains the file has been modified. If the two characters are hyphens (--) it means that the buffer was not modified. If these two characters are percent signs (%%) it means that the buffer cannot be modified (it is read-only). The name of the file being edited is shown in the mode line and if no file is being edited the word *scratch* is shown. The editor mode is shown in parenthesis. In this case the editor mode is (Text). Finally the word “All” refers to the amount of the file you can see on the screen. “All” means you can see the whole file. “Top” means you can see the top of the file. “Bot” means you can see the bottom of the file. If you can only see NN% of the file NN% will be shown.

The EMACS editor’s key-chords are somewhat different than other editors. There are some commands that require a single key-chord, for example CONTROL-a or CONTROL-e, but many commands require more than a single key-chord.

For example, to exit EMACS the key-chord sequence

CONTROL-x CONTROL-c

is used.

To save the file you are editing use the key-chord sequence

CONTROL-X CONTROL-S

If you exit EMACS before saving the file, EMACS will ask you if you want the file to be saved.

The key-chord for help is CONTROL-H. Figure 5.2 depicts the EMACS help screen.

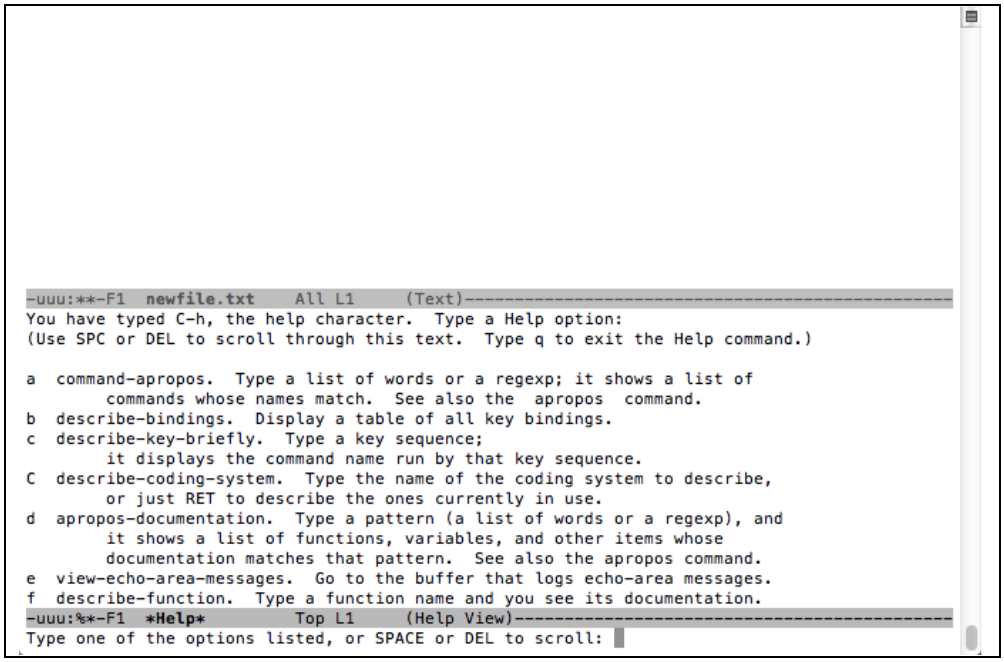

-uuu:**-F1 newfile.txt   All L1   (Text)
You have typed C-h, the help character. Type a Help option:
(Use SPC or DEL to scroll through this text. Type q to exit the Help command.)
a command-apropos. Type a list of words or a regexp; it shows a list of
commands whose names match. See also the apropos command.
b describe-bindings. Display a table of all key bindings.
c describe-key-briefly. Type a key sequence;
it displays the command name run by that key sequence.
C describe-coding-system. Type the name of the coding system to describe,
or just RET to describe the ones currently in use.
d apropos-documentation. Type a pattern (a list of words or a regexp), and
it shows a list of functions, variables, and other items whose
documentation matches that pattern. See also the apropos command.
e view-echo-area-messages. Go to the buffer that logs echo-area messages.
f describe-function. Type a function name and you see its documentation.
-uuu:%*-F1 *Help*   Top L1   (Help View)
Type one of the options listed, or SPACE or DEL to scroll:

Figure 5.2 Help screen displayed when you enter CONTROL-H

When you enter an EMACS command it is possible to undo that command with the key-chord CONTROL-_ (underscore). One of the interesting features of EMACS is that the undo is infinite, meaning that you can undo as far back as when you started the editing session.

In addition to the key-chord command prefix CONTROL-x there is also another special sequence called a META sequence. A META sequence begins with an ESCAPE. The ESCAPE key is called the META key.

Text is entered directly into the editor window. The normal operation of the editor allows text to be entered at any time.

Table 5.1 below summarizes common operations and their key-chords (or keys).

Supposing you delete some text but you want it back. The text that is deleted is held in a place called the KILL buffer. To retrieve the deleted text, the key-chord CONTROL-y can be used. The Y stands for yank.

Command	Description
CONTROL-X CONTROL-S	Save the file
CONTROL-X CONTROL-C	Exit EMACS
CONTROL-A	Move to the beginning of the line
CONTROL-E	Move to the end of the line
CONTROL-D	Delete the character to the right of the cursor
DELETE	Delete the character to the left of the cursor
META-d	Delete the word to the right of the cursor
META-DELETE	Delete the word to the left of the cursor
CONTROL-K	Delete from the cursor to the end of the line
CONTROL-Y	Paste deleted text at the cursor

Table 5.1 Summary of EMACS Key-chords and Command Keys

VI

The VI editor is one of the original editors developed for UNIX. Originally it was strictly a character editor but over the years it has been extended to include commands to manipulate lines of text.

The VI editor (the VI editor program) is started with the UNIX command *VI*. At the command prompt enter the command,

vi filename

To create a file named “myprog.cpp” you would use the command,

vi myprog.cpp

This command will run the VI editor program to create the file named “myprog.cpp”. The filename extension of .cpp indicates this file is a C++ program file.

If you are going to create a new file with the VI editor, when VI shows the new file each line begins with a tilde(~). This signifies that the line has nothing on it. This is shown in Figure 5.3.

Figure 5.3 – The opening VI display

Notice a couple of things about the opening VI display. First, the small square block at the beginning of the screen is called the cursor and it shows where in the file manipulations will occur. In other words, this is the "current position" in the file. The tildes (~) on each line mean that the line is empty – it has no characters in it. At the bottom of the display the current filename is shown along with an indication of whether the file is new. At this point you can begin using VI to create the text file.

The VI editor has two important modes. An editor mode determines the way that the editor is programmed to operate. One of the modes of the VI editor is called "command mode" and the other mode is the "character mode."

In command mode the editor accepts commands and performs whatever it is that the command instructs VI to do. For example, supposing you were using VI and you wanted to return to the UNIX command-prompt. To accomplish this you would use the command ":q" (a colon followed by the lower-case letter q), as shown in Fig. 5.4

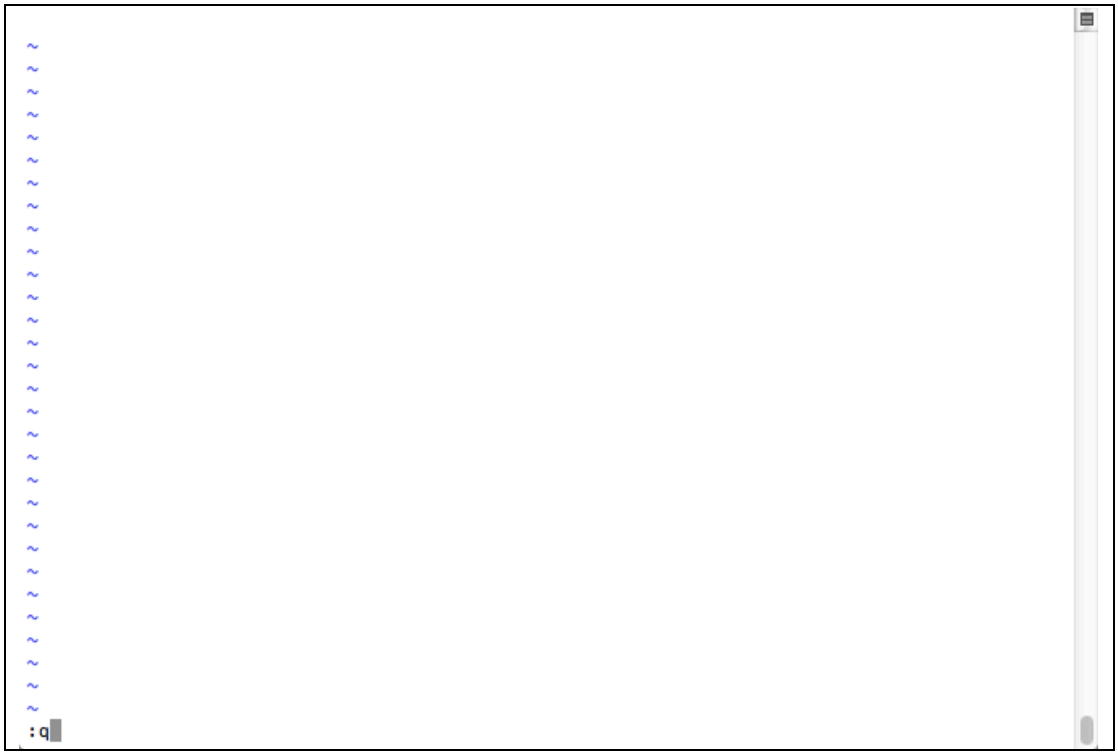


Figure 5.4 Entering the "quit" command (:q)

After you have typed the command (:q) you press the "enter" key and VI will attempt to execute the command. If VI understands the command it will do what it is told as is shown in Figure 5.5.

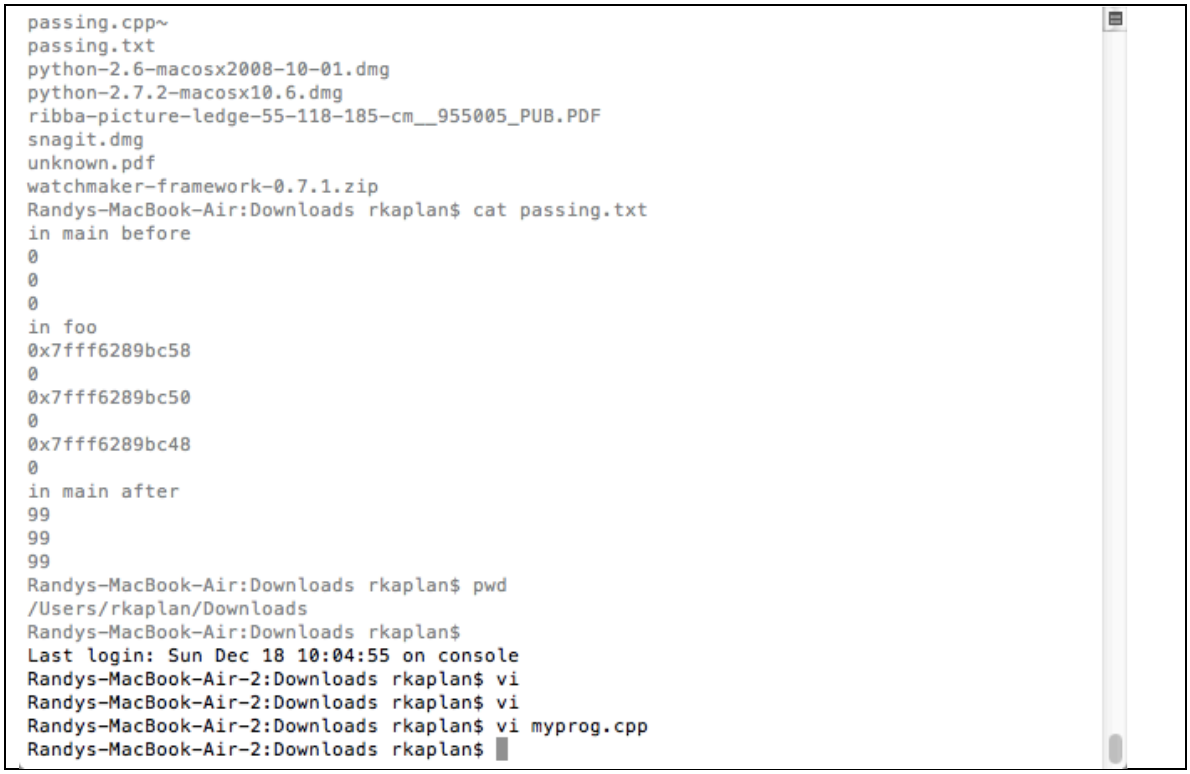


Figure 5.5 After the command :q is entered

Notice that the editor display is gone and we are now at the command prompt again.

In character mode you are able to enter, change, and delete characters in the file you are editing. For example, suppose you use the insert command. Once you enter this command the editor chooses as its current position wherever the cursor is positioned. At this point you can type any text and it will appear on the screen. As you type text and fill a line, the tilde(~) will disappear. See figure 5.6.

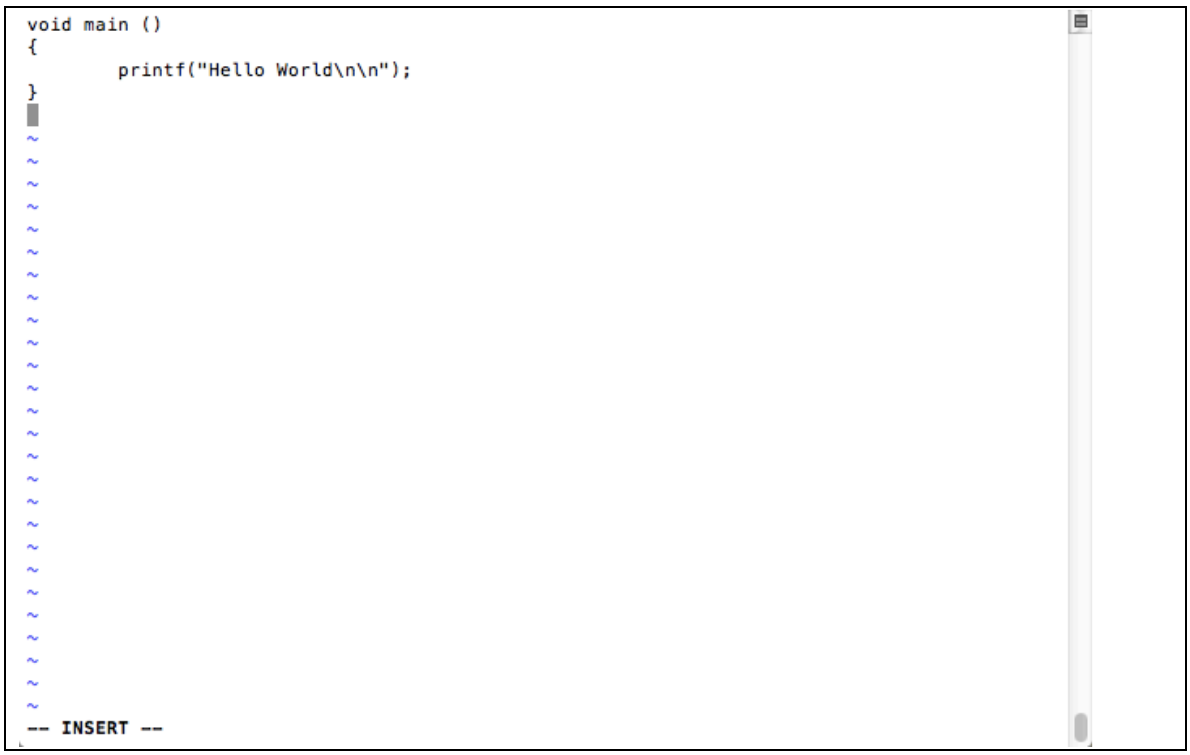


Figure 5.6 Entering some program text in insert mode

As you can see in this figure, a small program has been entered. It was entered by simply typing the text on the keyboard. As you type text, the text appears on the screen. The VI editor at this point is still in character mode. To switch back to command mode so you can, for example, save the file, you can enter or press the escape ("esc") key. Once the escape key is pressed you will return to command mode, as shown in Fig 5.7.

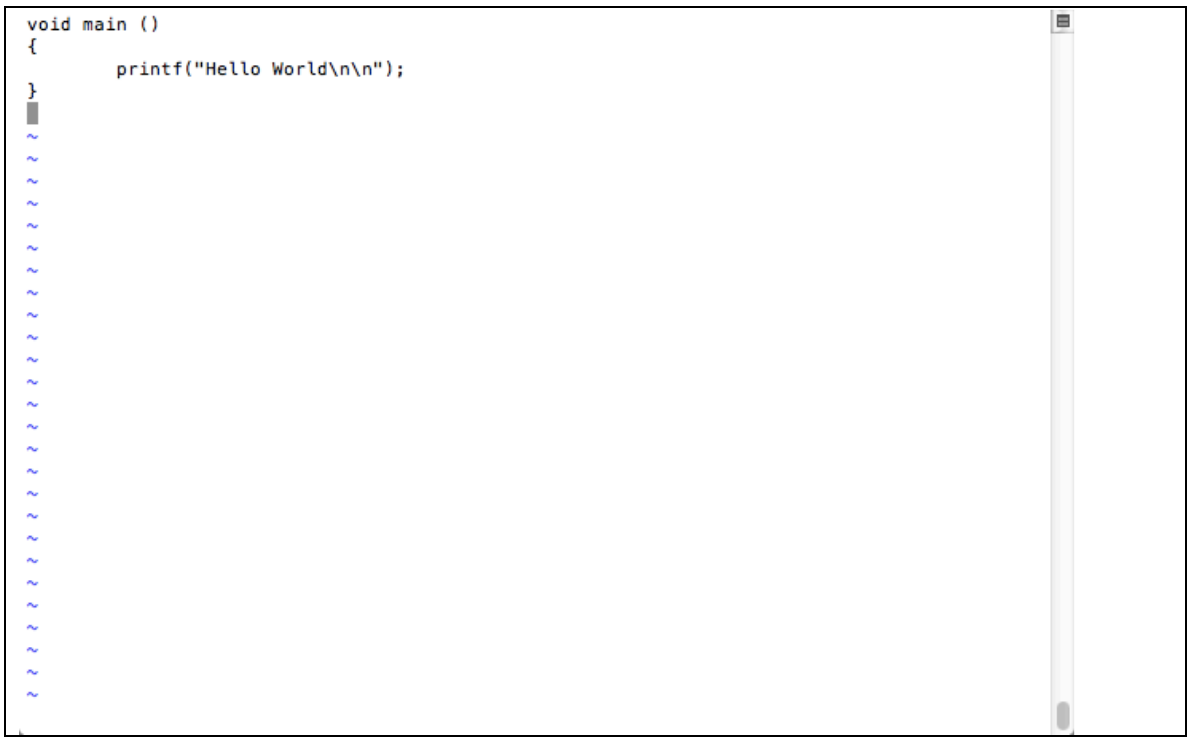


Figure 5.7 – VI editor screen showing a brief file

Notice that the mode indicator at the bottom of the screen has disappeared. This means you have returned to command mode.

To save the file enter the command :w (colon w), as shown in Fig 5.8.

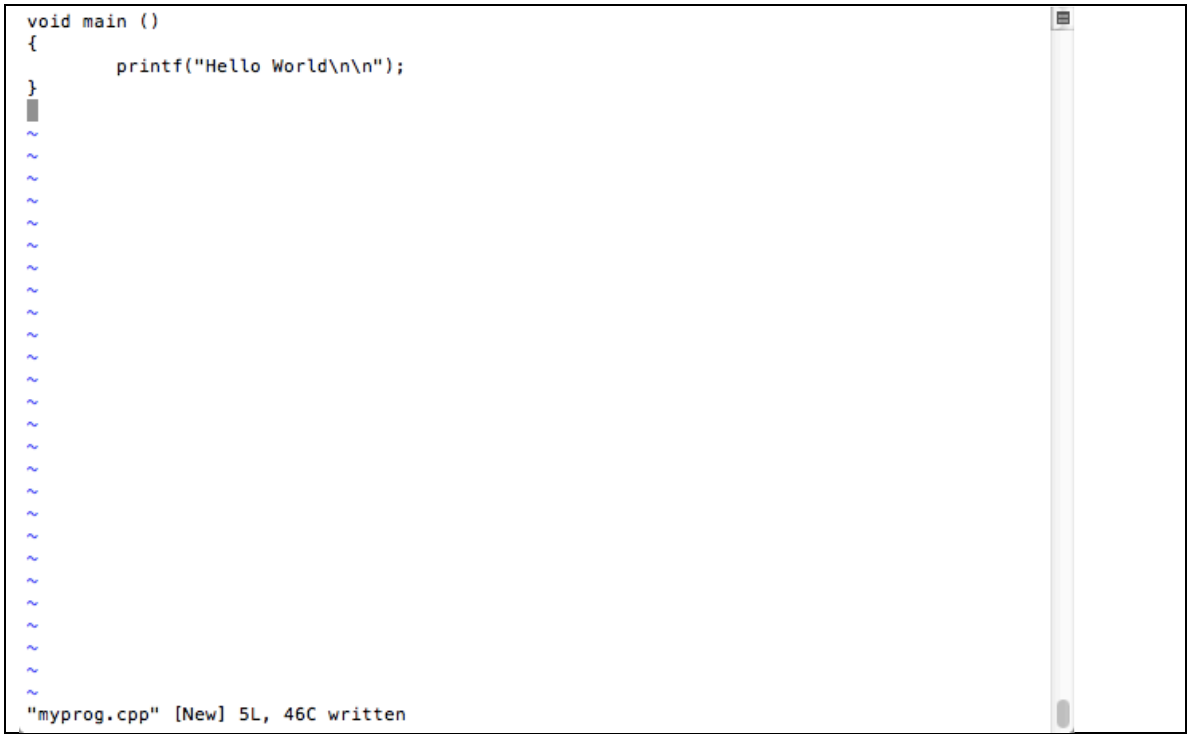


Figure 5.8 Saving a file

Basic editing commands of VI are summarized in the next table.

Command	Description
:q	exit from VI
:w	Write the file to the disk drive
I	Insert at the cursor
X	Delete at the cursor
U	Undo the preVIous action
A	Append (add new) text at the cursor
O	Open a line at the cursor

Table 5.1.2.1 Summary VI Commands

Of course entering text is not the only purpose an editor can serve. It is also important to understand how to add text to the file, replace text in the file, delete text in the file, or undo changes that have been made to the file.

PICO

The PICO editor is another editor provided for users at KU. It is a full screen editor meaning that the whole screen is used to display the contents of a file. It is considered to be a line editor (as opposed to a character editor). The PICO editor is started with the command

pico filename

Fig 5.9 shows the PICO editor display.

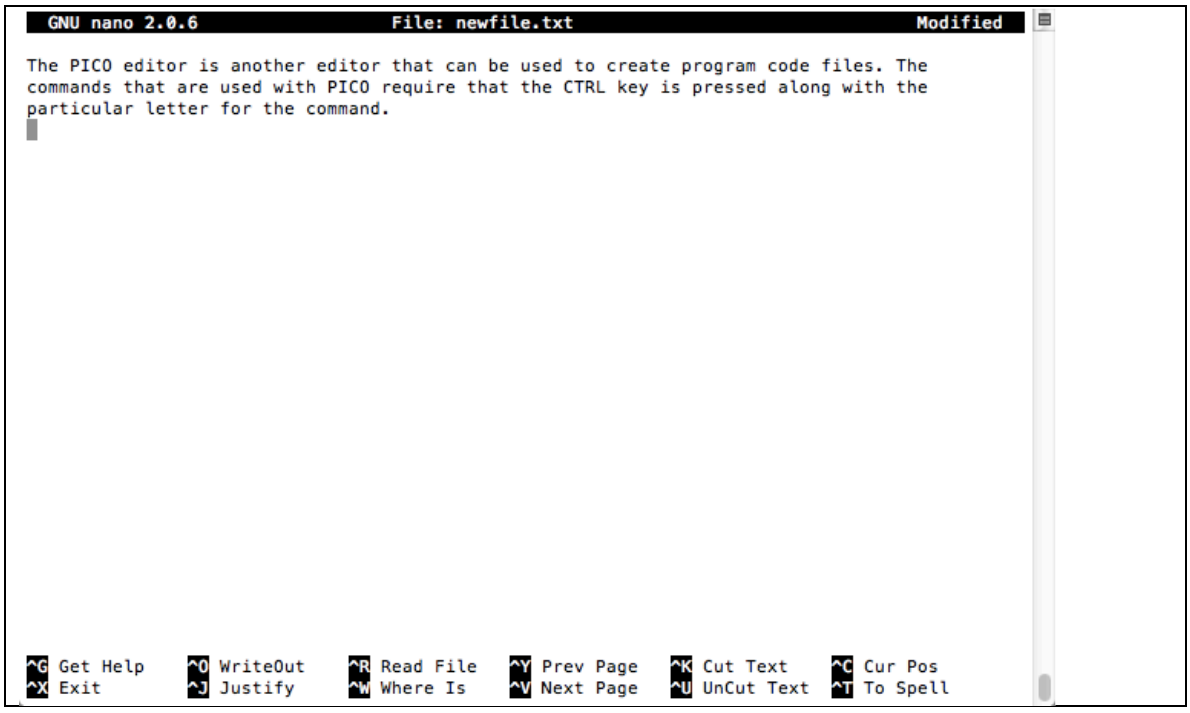

GNU nano 2.0.6 File: newfile.txt Modified
The PICO editor is another editor that can be used to create program code files. The commands that are used with PICO require that the CTRL key is pressed along with the particular letter for the command.
^G Get Help ^O WriteOut ^R Read File ^Y Prev Page ^K Cut Text ^C Cur Pos
^X Exit ^J Justify ^W Where Is ^V Next Page ^U UnCut Text ^T To Spell

Figure 5.12 PICO display

At the bottom of the PICO display you can see a legend that specifies the common commands. The karat(^) denotes the CONTROL key.

Cursor movement in PICO is accomplished by using the standard cursor keys. There are some useful key-chords that are worth noting. A summary of the commands are shown in Table 5.1.3.1.

Command	Description
Control-A	Move to the beginning of a line
Control-E	Move to the end of the line
Control-N	Move to the next line
Control-P	Move to the previous line
Control-O	Save the file
Control-X	Exit the editor

Table 5.1.3.1 – Summary of Useful PICO commands

Debuggers

A debugger is a utility that facilitates program debugging. Debugging is a process of observing the behavior of a program as the program is running. Observing the behavior of a program requires that you be able to stop the program at certain points so you can examine the value of variables. Also it allows the program to be executed statement by statement. At any step of the program it is possible to examine variables at this point. If the program does not do what is expected, corrections can be made to the program and it can be re-run.

In order to debug a program it is necessary to tell the compiler that you are going to do so. The option –g is used for this purpose. The compile command should be –

g++ -g myprogram.cpp

This command will make the program ready to be run in the debugger.

Unless you specify otherwise the compiler produces an executable file named a.out. To use the GNU debugger with a.out you would enter the command –

gdb a.out

Once you enter the command, the debugger loads with your program. Your program does not start running until you tell the debugger to run your program. The command to accomplish this is –

run

Prior to starting a debugging session with run, it is important to set up one or more breakpoints. A breakpoint is a place in a program where the program will stop so you can examine the state of variables at that point in the program. Breakpoints are specified with the **break** command.

There are a couple of ways that you are able to specify the location of a breakpoint. You can specify the location of a breakpoint by specifying the line of the program where you want it

to stop. Line numbers are just numbers that identify a particular line in the program. For example if you wanted to stop at line 20 you would use the break command as follows –

break 20

Another way to specify a breakpoint is to specify the name of a function to break at. In other words if you want to stop the program when a particular function is executed you can set a breakpoint at that point by using the function's name. Suppose you have a function named *addOne*. To stop the program when this function is called you use the break command as follows –

break addOne

When you run the program in the debugger with the run command it will run up until the first break point. When the program stops at the break point there are several things that you are able to do. Suppose that you had another break point and you wanted to see what would happen when the program reached the second break point. In order to accomplish this, use the command "continue."

If you wish to step through the program one line at a time you can use the command "step." The step command steps through every statement. When a function is encountered the debugger will execute each line in the function in a step-by-step manner. If you wish to execute the program as if a function call is a single instruction use the command "next." The next command does not descend into the function. Other than this difference the next and step commands operate in the same way.

If you need to enter the step command or the next command a number of times you can do so by pressing the ENTER key after the first time you entered the command. The debugger will repeat the last command as many times as necessary.

As mentioned earlier, another important aspect of debugging is the ability to examine values of variables in the program at the particular line where the program has stopped. The debugger command that displays the value of a variable is –

print variableName

where variableName is the name of the variable whose value you wish to display. Sometimes it is useful to be able to display a variable's value in its hexadecimal form. To do this there is a variant of the print command –

print/x variableName

A watchpoint acts on variables. Setting a watchpoint for a variable causes the debugger to stop the program when the value of that variable changes. In order to set a watchpoint for the variable named total the command –

watch total

would be used.

When a breakpoint is set it is assigned a number. This number is used to refer to the breakpoint. For instance it is sometimes desirable to delete a breakpoint. A breakpoint can be deleted by using the command –

delete 5

which would delete the breakpoint assigned 5.

Makefiles

Sometimes it makes sense to automate the compilation of a program. This is especially true if the program is composed of many smaller parts – say objects– one object to a file. When making changes in such a program some of the files may need to be re-compiled and some may not need to be re-compiled. This can become quite tedious if there are many files that make up the program. The makefile is a way to automate this process. A utility named "make" is used to accomplish the task of building an executable automatically.

Note: This is just an introduction to makefiles and the utility named make – actually GNUMAKE which is the Free Software Foundation version of make. Makefiles can become very complex because in essence a makefile is like a program that tells the make utility how to compile a program. Once you have mastered the basics of make files, more sophisticated capabilities can be learned to take advantage of makefile features.

A make file consists of a series of rules. Each of the rules has the form –

target …. prerequisites …
rule
…
…

A target is the name of a file that you wish to build. It can be any file but it is usually an object file or an executable file. A prerequisite is a file used in the construction of the target. You could think about prerequisites as files that are input needed to create the target. The rule consists of one or more commands used to create the target. A TAB must precede each rule command. This helps the make utility to recognize the recipe commands.

How does the make utility know what to do? In other words when building a particular target how does the make utility decide if the target needs to be made? This is determined by the prerequisites. If any of the prerequisites have changed since the previous build, make will follow the recipe (series of commands) to rebuild the target.

Here is an example of a simple make file. It is best to name the make file *makefile* or *Makefile.*

```
edit : main.o kbd.o command.o display.o \
        insert.o search.o files.o utils.o
         cc -o edit main.o kbd.o command.o display.o \
                 insert.o search.o files.o utils.o
   main.o : main.c defs.h
         cc -c main.c
   kbd.o : kbd.c defs.h command.h
         cc -c kbd.c
   command.o : command.c defs.h command.h
         cc -c command.c
   display.o : display.c defs.h buffer.h
         cc -c display.c
   insert.o : insert.c defs.h buffer.h
         cc -c insert.c
   search.o : search.c defs.h buffer.h
         cc -c search.c
   files.o : files.c defs.h buffer.h command.h
         cc -c files.c
   utils.o : utils.c defs.h
         cc -c utils.c
   clean :
         rm edit main.o kbd.o command.o display.o \
           insert.o search.o files.o utils.o
```

To use this file to create the executable file named “edit” use the command –

```
make
```

If the make file is named something other than *makefile* or *Makefile*, then you must specify the name of the makefile after the *make* command.

To use this file to erase the executable file and all of the object files use the command –

```
make clean
```

Note: The file named makefile must be in your current directory. The make utility looks for a file named makefile or Makefile and automatically processes its contents.

When a target is a file, it will be recompiled or relinked whenever any of its prerequisites change. In turn, any of the perquisites on which the target depends that are automatically generated should be updated prior to updating the target file. In this example the target is dependent upon eight files.

main.o
kbd.o
command.o
display.o
insert.o
search.o
files.o
utils.o

Looking further in the makefile we see that each of the prerequisites of the first target have their own rules. In all cases the prerequisite contains a source file that corresponds to the target prerequisite object file. In some cases the prerequisites of the targets on which the first target depends include header files. A change to any of these files will cause the corresponding "sub-target" to be rebuilt.

In the example makefile, some of the lines have a backslash (\) at the end of the line. The backslash designates a continuation of the line. In other words, this character is used when you want to continue a line in the makefile on another line.

One of the ways to simplify a makefile is to make use of variables. A variable can be set to a file name or a list of file names that can be used as shorthand for that list. It is standard practice to have a variable named OBJECTS (there are many variants of the name) that names the objects on which the target is dependent. Here is an example of a variable set for the example makefile.

OBJECTS = main.o kbd.o command.o display.o insert.o search.o files.o utils.o

To use this variable in a makefile the notation is –

$(OBJECTS)

So using the variable would change the example makefile as show next.

```
OBJECTS= main.o kbd.o command.o display.o \
         insert.o search.o files.o utils.o
   edit : $(OBJECTS)
        cc -o edit $(OBJECTS)
   main.o : main.c defs.h
        cc -c main.c
   kbd.o : kbd.c defs.h command.h
        cc -c kbd.c
   command.o : command.c defs.h command.h
        cc -c command.c
   display.o : display.c defs.h buffer.h
        cc -c display.c
   insert.o : insert.c defs.h buffer.h
        cc -c insert.c
```

```
search.o : search.c defs.h buffer.h
        cc -c search.c
files.o : files.c defs.h buffer.h command.h
        cc -c files.c
utils.o : utils.c defs.h
        cc -c utils.c
clean :
        rm edit $(objects)
```

It seems repetitive to have to keep writing the same command for compiling a file over and over again. Why can't make be smart enough to be able to look at the file type and "figure out" how to create the file? Well in fact it can. Look at the third version of the example makefile.

```
OBJECTS= main.o kbd.o command.o display.o \
         insert.o search.o files.o utils.o
edit : $(OBJECTS)
        cc -o edit $(OBJECTS)
main.o : defs.h
kbd.o : defs.h command.h
command.o : defs.h command.h
display.o : defs.h buffer.h
insert.o : defs.h buffer.h
search.o : defs.h buffer.h
files.o : defs.h buffer.h command.h
utils.o : defs.h
```

Notice now that instead of specifying each compile command we simply name an object file (having a corresponding .c file) and its dependencies. The make utility is smart enough to understand how to build the object file from the source file, e.g. kbd.c is compiled to create kbd.o.

Afterword

All of the information presented in this chapter is meant only to be an introduction to using the programs that have been described. All of these programs have significantly more features than described here and there may be times when you want to use some of these more advanced features. All of the information contained in this chapter was derived from resources on the World Wide Web. There are tremendous resources for anything "programming" on the World Wide Web and if you should have a question you should be able to get it answered by accessing these resources. For example, for each of these programs I used the Google search string "xxxxxx tutorial" where xxxxxxx is the name of the program for which I wanted the tutorial. For example if I wanted to find a tutorial written for GNUMAKE I would enter "GNUMAKE tutorial" as the search string in Google. This would result in hundreds of links to tutorial resources for GNUMAKE.

Chapter 6 Computer Science Curriculum

Computer Science Degrees

The Computer Science Department offers two degrees: a Bachelor's Degree in Computer Science and Master's Degree in Computer Science. The Bachelor's Degree offers two tracks, a Software Development track and an Information Technology track. The Master's Degree offers three tracks, a Software Development track, an Information Technology track and a Professional Science Master's (PSM) track. There is also a minor offered in Computer Science. The minor also has two tracks, Software Development and Information Technology.

The Software Development (SD) track focuses the student's study on application development and theoretical concepts. The career of graduates of this track typically involves designing and implementing software, devising new ways to use computers or developing effective ways to solve computing problems.

The Information Technology (IT) track focuses the student's study on topics in web development, networks, databases, and security. The emphasis in these courses is more on the technology itself. Employees in the IT field typically provide the computing services required to support the organization's mission, such as install and manage the network, design web pages, administrate the systems, and plan and manage the technology lifecycle of the organization's technology.

When declaring Computer Science as a major, you must determine the track you would like to pursue. It is possible to change tracks, but you should determine your track for graduation as early as possible, but definitely by the time you begin your senior year (when you reach 90 credits).

Five-year Program

The Computer Science Department offers a five-year program. This program allows a student to obtain a Bachelor's Degree and a Master's Degree in Computer Science in five years. This timeframe is based on the completion of 15 new credits each semester for the first four and then 9 new credits each semester for the fifth year. This is also based on the student completing 12 credits of 400-level CSC courses in the free electives section of the general education portion of the program. These 12 credits of 400-level courses count toward the required credits in both the Bachelor's and Master's degree. The student will receive the Bachelor's Degree upon completion of those degree requirements and then will receive the Master's Degree upon completion of those degree requirements.

Professional Science Master's Degree

The Professional Science Master's (PSM) track is an innovative graduate degree designed to allow you to pursue advanced training and excel in science while simultaneously

developing highly-valued business skills. PSM programs prepare you for science careers in business, government, or nonprofit organizations, where workforce needs are increasing. Programs are characterized by "science-plus," combining rigorous study in science or mathematics with skills-based coursework in management, policy, or law. PSM programs emphasize writing and communication skills. The PSM includes a final project or team experience, as well as a "real-world" internship in a business or public sector enterprise.

Degree Requirements
Special Requirements

The following specifies the requirements for the Computer Science degrees. This includes requirements for course prerequisites, repeating courses, and graduation requirements. Some of the requirements are defined by the university and some are defined by the Computer Science Department.

Course Prerequisites

In order to take any CSC course numbered 300 or above, a student must meet the following two requirements:

- A GPA of at least 2.25 in CSC courses and a 2.0 GPA overall
- Completed 18 semester hours of computer science courses numbered 125 or above

In addition to this, two courses have specific credit hour prerequisites. To register for CSC354, a student must have completed 24 semester hours of CSC courses numbered 125 or above. To register for CSC385, a student must have completed 36 semester hours of CSC courses numbered 125 or above. All CSC majors must obtain a grade of 'C' or better in CSC135 and CSC136.

There are also university requirements for taking a 400-level course, which are considered graduate-level courses, for undergraduate students. Enrollment in 400-level courses is limited to junior, senior, and graduate students. Undergraduate students at the junior (at least 60 credits) or senior (at least 90 credits) with less than a 2.9 cumulative grade point average must obtain the permission of the instructor and dean.

Course Repeats

All Undergraduate students will be limited to a maximum of six course repeats total at Kutztown University. A single course can be attempted a maximum of three times (repeated twice). The grade from the last attempt of a course (regardless of whether it is higher or lower than previous attempts) will be the grade used for GPA calculation. If you fail a required course in your last attempt of the course, you MUST repeat the course. If that attempt is the sixth repeat, you will need to have an undergraduate exception petition approved. This is not guaranteed and you may be unable to graduate.

Graduation Requirements

In order to graduate with a Computer Science degree, you must have a GPA of 2.25 in all CSC courses and a cumulative GPA of 2.25 to graduate. All undergraduate students seeking their first baccalaureate degree are required to take at least 30 of their last 45 credits and at least 50% of the credits in their major at Kutztown University.

Permission to take a 500-level course

Undergraduates are permitted to take 500-level courses under specific circumstances. An eligible student may take a 500-level course for credit towards a graduate degree; a 500-level course may NOT count toward meeting the requirements of any Bachelor's degree. The student must be a graduating senior and must have a minimum GPA of 3.0. At the time of registration, this choice must be made and is not subject to change. Requests to take 500-level courses require specific forms, which are available on the Graduate Studies web site (http://www.kutztown.edu/academics/graduate/). The request requires by the professor, advisor, Department Chairperson, Dean of the College, and Dean of Graduate Studies.

Checksheets

The current checksheets for the Computer Science programs are included in Appendix A. There is a sequence of courses, some due to prerequisites, that is recommended for students. These course recommendations can be found in Appendix B.

Appendix C contains a tentative two-year rotation cycle for course offerings. Please note that this is subject to change based on faculty resources and current student needs.

Computer Science Careers

KU's Career Development Center publishes a "Major Discovery Series", which provides information about the majors offered at Kutztown University. This includes information such as possible careers in the field. The information sheets for the Computer Science degrees can be found in Appendix D.

Chapter 7 Student Organizations Related to Computer Science

Introduction

Joining a computing-oriented student organization, attending its meetings, and contributing to its activities are great ways to get to know your peers outside of the classroom, to broaden your knowledge of computing, to create programs and systems that are useful to you, and to improve your resume.

The Kutztown Technology Association[2]

The **KTA** is the primary computing-oriented student organization at Kutztown. It exists to promote technology in a friendly and meaningful way, centering on current technology and computer systems. The 2011-2012 KTA web site has this to say about the organization:

> *"The KTA has professional speakers come in to talk about their areas of expertise. Not only will you get to learn from a seasoned professional, but you'll be given the opportunity to engage in networking activities and gain strong connections.*
>
> *We hold workshops to train and give hands on experience on certain subjects. We are known for our patch cable making workshops, and will be launching a Build a Computer Workshop this year.*
>
> *We also host large gatherings of 20+ professionals from the AITP, where KTA student members are given the opportunity to network with a wide array of professionals. The most recent event of this magnitude was our first Microsoft Day.*
>
> *When we're not hosting an event of these types, we hold a weekly meeting. During these meetings we either give a presentation on current and important topics to the technological field, hold discussions, or engage in debates."*

The KTA is a student branch of the Association of Information Technology Professionals (AITP). Since 1951, the AITP has been a continuing resource for quality professional development. Serving the IT industry as a national organization and through our local chapters, AITP is committed to delivering to our members the recognition, support, and credibility that will give them the competitive edge needed to succeed. By joining AITP, you become part of a strong network of IT professionals, have access to empowering resources and opportunities, and are positioned to achieve your professional goals.

[2] http://unixweb.kutztown.edu/kta/

The KTA elects new officers each year who determine the activities that will take place during the year and also run the weekly meetings. Dr. Joo Tan (tan@kutztown.edu) is the faculty advisor for KTA. KTA sends out regular notices about meeting schedules and planned presentations. Participating in KTA is one of the best ways to make friends with people pursuing the same major and to get the low-down from veteran CSC students.

The Mobile Apps Development Club

MAD was formed in the fall of 2011 by Mike Cutalo (mcuta697@live.kutztown.edu) and Tony Setting (tsett075@live.kutztown.edu) to explore and promote the development and use of applications for mobile devices including the iPhone, iPad, and Android devices. The emphasis of the club is planning, developing and deploying mobile apps. The 2011-2012 project for both platforms is a mobile assistant for Kutztown students with maps, links, and built-in knowledge for navigating the complex geographic and social environment of Kutztown University and the town of Kutztown. Some MAD meetings include technical talks from faculty and students working on other mobile projects. The faculty advisor is Dr. Dale Parson (parson@kutztown.edu).

The Kutztown University Programming Team

In 2009 the Computer Science Department formed the Kutztown University Competitive Programming Team. The team is composed primarily of computer science majors competing with other Universities in the region. Competitions consist of writing programs in a limited time period that solve specified problems correctly. The team that solves the most problems in the shortest time wins the competition. The Kutztown team meets on Friday afternoons at 3 PM. Practices consist of small teams working on problems from previous competitions. Twice a year the team attends competitions. The Programming Team is one way that students can improve their overall programming skills. The faculty advisor is Dr. Randy Kaplan.

Appendix A

Computer Science Degree Checksheets

This Appendix contains copy of the checksheets for each Computer Science major and minor. Please note that the checksheets change frequently so you should check the CS web site (http://cs.kutztown.edu) or the department office to ensure you are working with the most recent version.

STUDENT: _____ STUDENT ID NUMBER: _____

KUTZTOWN UNIVERSITY

COLLEGE OF LIBERAL ARTS & SCIENCES • BS • SD

Program Number: 21621 **Version Number:** 20123 **Effective Date:** Fall 2011

GENERAL EDUCATION

I. UNIVERSITY CORE (12 credits)	RC	CR	GR
A. Oral Communication: SPE 010 or above			
COURSE:	3		
B. Written Communication: ENG 023, 024, or 025			
COURSE:	3		
C. Mathematics: MAT 017 or above			
COURSE: MAT 140 or 301	3	**3**	
D. Wellness: HEA 110 + activity; or 3-credit Wellness course			
COURSE:	3		

II. UNIVERSITY DISTRIBUTION (15 credits)	RC	CR	GR	CAC
A. Natural Sciences: Any lab or non-lab course with prefix AST, BIO, CHM, ENV, GEL, MAR, or PHY; or certain GEG courses (see note at right)				
COURSE:	3			
B. Social Sciences: Any course with prefix ANT, CRJ, ECO, HIS, INT, MCS, PSY, POL, SOC, or SWK; or certain GEG courses (see note at right)				
COURSE:	3			
C. Humanities: Any course with prefix ENG, HUM, PAG, PHI, WRI, WST, or Modern Language				
COURSE: WRI 205	3	**3**		
D. Arts: Any course with prefix ARC, ARH, ART, CDE, CDH, CFT, DAN, FAR, FAS, MUP, MUS, or THE				
COURSE:	3			
E. Free Elective: Any course carrying university credit				
COURSE:	3			

III. COMPETENCIES ACROSS THE CURRICULUM	RC	CR	GR	CAC
A. Writing Intensive (**WI**) (9 credits)				
COURSE:	3			**WI**
COURSE:	3			**WI**
COURSE:	3			**WI**
B. Quantitative Literacy (**QL**) (3 credits) OR Computer-Intensive (**CP**) (3 credits)				
COURSE:	3			
C. Visual Literacy (**VL**) (3 credits) OR Communication-Intensive (**CM**) (3 credits)				
COURSE:	3			
D. Cultural Diversity (**CD**) (3 credits)				
COURSE:	3			**CD**
E. Critical Thinking (**CT**) (3 credits)				
COURSE:	3			**CT**

A Competency Across the Curriculum (CAC) course is not a separate course, but rather an overlay that is "double counted" as fulfilling both the CAC requirement and another requirement in either General Education (except for the University Core), the major, or the minor.

RC = Minimum required number of credits
CR = Credits earned (fill in number of credits)
GR = Grade earned (fill in letter grade)
CAC = Competency Across the Curriculum (fill in designation)

NOTE: GEG courses with a lab and 040, 322, and 323 may be used in II.A. and GEG courses 040, 204, 274, 305, 322, 323, 324, 347, 380, and 394 may NOT be used in II.B.

IV. COLLEGE DISTRIBUTION (33 credits)	RC	CR	GR	CAC
A. Natural Science, Mathematics, and Computer Science# (6 credits): Choose one course in each subcategory.				
1. Natural Science with Lab: AST, BIO, CHM, ENV, GEL, PHY, or MAR; or GEG (see note at right)				
COURSE:	3			
2. Elective: MAT, CSC, AST, BIO, CHM, ENV, GEL, PHY, or MAR; or GEG (see note at right)				
COURSE:	3			
B. Social Science (9 credits): Choose one course in each subcategory.				
1. Elective: HIS, ANT, GEG (see note at right), or POL				
COURSE:	3			
2. Elective: PSY, SOC, CRJ, or SWK				
COURSE:	3			
3. Elective: ANT, HIS, ECO, GEG (see note at right), PSY, POL, SOC, CRJ, or SWK				
COURSE:	3			

	RC	CR	GR	CAC
C. Humanities (9 credits): Choose one course in each subcategory.				
1. Elective: PAG*, ENG, WRI, or HUM				
COURSE:	3			
2. Elective: (MLS, GER, SPA, FRE, CHI, or ARA)* or PHI				
COURSE: PHI 40	3	**3**		
3. Elective: (PAG, MLS, GER, SPA, FRE, CHI, or ARA)*, ENG, WRI, HUM, or PHI				
COURSE:	3			
D. Free Electives (9 credits): Choose any university courses that count toward graduation.				
COURSE:	3			
COURSE:	3			
COURSE:	3			

NOTE: GEG courses with a lab and 040, 322, and 323 may be used in IV.A. and GEG courses 040, 204, 274, 305, 322, 323, 324, 347, 380, and 394 may NOT be used in IV.B.

Students in the College of Liberal Arts and Sciences are required to take at least one course in Biological Science (BIO) and at least one course in Physical Science (AST, CHM, ENV, GEL, PHY, MAR, GEG with lab, or GEG 040, GEG 322, or GEG 323), and at least one of which must be a lab (each course may be counted in either sections II.A or IV.A).

* Excludes courses in language instruction, conversation, composition, linguistics, syntax, stylistics, and internship courses; contact the Modern Language Studies Department for clarification.

B. Major Program: 51 sh		
1. Required Courses: 33 sh	Gr	SH
CSC 125: Discrete Math for CS I		
CSC 225: Discrete Math for CS II		
CSC 135: Comp Sci I		
CSC 136: Comp Sci II		
CSC 235: Comp Org & Asm. Lng.		
CSC 237: Data Structures		
CSC 310: Prog. Languages		
CSC 325: Intro to CS Theory		
CSC 343: Operating Systems		
CSC 354: Software Engineering		
CSC 385: Seminar in CSC		
2. Elective Courses: 18 sh (no more than two 200-level)	Gr	SH
CSC 241: Adv Visual Basic		
CSC 242: Web Programming		
CSC 243: Java Programming		
CSC 253: IT Systems		
CSC 311: Computer Networks		
CSC 320: Intro to Game Programming		
CSC 341: Information Security		
CSC 342: Web Technologies		
CSC 351: Digital Forensics		
CSC 352: UNIX: Sys Prog/Adm		
CSC 356: Introduction to Database Sys		
CSC 402: Data Structures II		
CSC 411: Advanced Networking		
CSC 421: Web-Based Soft Design & Dev		
CSC 425: Compiler Design I		
CSC 445: Intro to Intelligent Robotics		
CSC 447: Artificial Intelligence I		
CSC 456: Database Mgm Sys I		
CSC 464: Human Computer Interaction		
CSC 480: Special Topics		

C. Concomitant Courses: 9 sh		
1. Required Courses: 9 sh	Gr	SH
MAT 171: Calculus I		
MAT 260: Linear Algebra		
MAT (elective higher than 171, not 224)		
MAT 140 or 301: Applied Stat Methods*	X	X
WRI 205: Scientific Writing*	X	X
PHI 040: Intro to Ethics*	X	X
*accounted for in general education		
2. Internship – optional (free elective)	Gr	SH
CSC 280: Cooperative Internship I		3-6
CSC 380: Cooperative Internship II		3-6
Notes on BS in Computer Science		
Before taking any 300-level course a student must have completed 18 credit hours in CSC courses numbered 125 or above with a GPA of 2.25 in the CSC courses.		
Students minoring in Math should take MAT 301		
CSC-prefix courses below 125-level, CSC 130, CSC 280 and CSC 380 do not count toward the BS in Software Development		

Internal Transfer: 2.25 GPA needed
Graduation requirement: 2.25 GPA Overall, 2.25 GPA Major
Program Code: 21621
Version Number: 20123

Approved: _____
Date:_____

STUDENT: _____ STUDENT ID NUMBER: _____

KUTZTOWN UNIVERSITY

COLLEGE OF LIBERAL ARTS & SCIENCES • BS • COMPUTER SCIENCE: IT

Program Number: 21611 **Version Number:** 20123 **Effective Date:** Fall 2011

GENERAL EDUCATION

I. UNIVERSITY CORE (12 credits)	RC	CR	GR
A. Oral Communication: SPE 010 or above			
COURSE:	3		
B. Written Communication: ENG 023, 024, or 025			
COURSE:	3		
C. Mathematics: MAT 017 or above			
COURSE: MAT 140	3	**3**	
D. Wellness: HEA 110 + activity; or 3-credit Wellness course			
COURSE:	3		

II. UNIVERSITY DISTRIBUTION (15 credits)	RC	CR	GR	CAC
A. Natural Sciences: Any lab or non-lab course with prefix AST, BIO, CHM, ENV, GEL, MAR, or PHY; or certain GEG courses (see note at right)				
COURSE:	3			
B. Social Sciences: Any course with prefix ANT, CRJ, ECO, HIS, INT, MCS, PSY, POL, SOC, or SWK; or certain GEG courses (see note at right)				
COURSE:	3			
C. Humanities: Any course with prefix ENG, HUM, PAG, PHI, WRI, WST, or Modern Language				
COURSE: WRI 207	3	**3**		
D. Arts: Any course with prefix ARC, ARH, ART, CDE, CDH, CFT, DAN, FAR, FAS, MUP, MUS, or THE				
COURSE:	3			
E. Free Elective: Any course carrying university credit				
COURSE:	3			

III. COMPETENCIES ACROSS THE CURRICULUM	RC	CR	GR	CAC
A. Writing Intensive (**WI**) (9 credits)				
COURSE:	3			**WI**
COURSE:	3			**WI**
COURSE:	3			**WI**
B. Quantitative Literacy (**QL**) (3 credits) OR Computer-Intensive (**CP**) (3 credits)				
COURSE:	3			
C. Visual Literacy (**VL**) (3 credits) OR Communication-Intensive (**CM**) (3 credits)				
COURSE:	3			
D. Cultural Diversity (**CD**) (3 credits)				
COURSE:	3			**CD**
E. Critical Thinking (**CT**) (3 credits)				
COURSE:	3			**CT**

A Competency Across the Curriculum (CAC) course is not a separate course, but rather an overlay that is "double counted" as fulfilling both the CAC requirement and another requirement in either General Education (except for the University Core), the major, or the minor.

RC = Minimum required number of credits
CR = Credits earned (fill in number of credits)
GR = Grade earned (fill in letter grade)
CAC = Competency Across the Curriculum (fill in designation)

NOTE: GEG courses with a lab and 040, 322, and 323 may be used in II.A. and GEG courses 040, 204, 274, 305, 322, 323, 324, 347, 380, and 394 may NOT be used in II.B.

IV. COLLEGE DISTRIBUTION (33 credits)	RC	CR	GR	CAC
A. Natural Science, Mathematics, and Computer Science# (6 credits): Choose one course in each subcategory.				
1. Natural Science with Lab: AST, BIO, CHM, ENV, GEL, PHY, or MAR; or GEG (see note at right)				
COURSE:	3			
2. Elective: MAT, CSC, AST, BIO, CHM, ENV, GEL, PHY, or MAR; or GEG (see note at right)				
COURSE:	3			
B. Social Science (9 credits): Choose one course in each subcategory.				
1. Elective: HIS, ANT, GEG (see note at right), or POL				
COURSE:	3			
2. Elective: PSY, SOC, CRJ, or SWK				
COURSE:	3			
3. Elective: ANT, HIS, ECO, GEG (see note at right), PSY, POL, SOC, CRJ, or SWK				
COURSE:	3			

	RC	CR	GR	CAC
C. Humanities (9 credits): Choose one course in each subcategory.				
1. Elective: PAG*, ENG, WRI, or HUM				
COURSE:	3			
2. Elective: (MLS, GER, SPA, FRE, CHI, or ARA)* or PHI				
COURSE: PHI 40	3	**3**		
3. Elective: (PAG, MLS, GER, SPA, FRE, CHI, or ARA)*, ENG, WRI, HUM, or PHI				
COURSE:	3			
D. Free Electives (9 credits): Choose any university courses that count toward graduation.				
COURSE:	3			
COURSE:	3			
COURSE:	3			

NOTE: GEG courses with a lab and 040, 322, and 323 may be used in IV.A. and GEG courses 040, 204, 274, 305, 322, 323, 324, 347, 380, and 394 may NOT be used in IV.B.

Students in the College of Liberal Arts and Sciences are required to take at least one course in Biological Science (BIO) and at least one course in Physical Science (AST, CHM, ENV, GEL, PHY, MAR, GEG with lab, or GEG 040, GEG 322, or GEG 323), and at least one of which must be a lab (each course may be counted in either sections II.A or IV.A).

* Excludes courses in language instruction, conversation, composition, linguistics, syntax, stylistics, and internship courses; contact the Modern Language Studies Department for clarification.

B. Major Program: 57 sh		
1. Required Courses: 33 sh	Gr	SH
CSC 125: Discrete Math for CS I		
CSC 130: IT Fundamentals		
CSC 135: Comp Sci I		
CSC 136: Comp Sci II		
CSC 242: Web Programming		
CSC 253: IT Systems		
CSC 311: Computer Networks		
CSC 341: Information Security		
CSC 356: Introduction to Database Sys		
CSC 354: Software Engineering		
CSC 385: Seminar in CSC		
2. Elective Courses: 15-24 sh	Gr	SH
CSC 237: Data Structures		
CSC 241: Adv Visual Basic		
CSC 243: Java Programming		
CSC 342: Web Technologies		
CSC 343: Operating Systems		
CSC 351: Digital Forensics		
CSC 352: UNIX: Sys Prog/Adm		
CSC 411: Advanced Networking		
CSC 421: Web-Based Soft Design & Dev		
CSC 445: Intro to Intelligent Robotics		
CSC 447: Artificial Intelligence I		
CSC 456: Database Mgm Sys I		
CSC 464: Human Computer Interaction		
CSC 480: Special Topics		
3. Elective Courses: 0-9 sh	Gr	SH
CSC 225: Discrete Math for CS II		
CSC 235: Comp. Org & Assem. Lang		
CSC 310: Proc Oriented Prog Lang		
CSC 320: Intro to Game Programming		
CSC 325: Intro to CS Theory		
CSC 402: Data Structures II		
CSC 425: Compiler Design I		

C. Concomitant Courses: 3 sh		
1. Required Courses: 3 sh	Gr	SH
MAT 105: College Algebra or above		
MAT 140: Applied Stat Methods*	X	X
WRI 207: Writing for Workplace*	X	X
PHI 040: Intro to Ethics*	X	X
*accounted for in general education		
2. Internship – optional (free elective)	Gr	SH
CSC 280: Cooperative Internship I		3-6
CSC 380: Cooperative Internship II		3-6
Notes on the BS Information Tech		
Consider taking a Minor in an Application Domain such as Math, Psychology, Sociology, Economics, Biology, or any Science		
Consider taking a second speech course in II E		
CSC-prefix courses below 125-level, CSC 280 and CSC 380 do not count toward the BS in Information Technology		
Before taking any 300-level course a student must have completed 18 credit hours in CSC courses numbered 125 or above with a GPA of 2.25 in the CSC courses.		

Internal Transfer: 2.25 GPA needed
Graduation requirement: 2.25 GPA Overall, 2.25 GPA Major
Program Code: 21611
Version Number: 20123

Approved: _____
Date: _____

STUDENT: _____ STUDENT ID NUMBER: _____

KUTZTOWN UNIVERSITY

COLLEGE OF LIBERAL ARTS & SCIENCES • BS/MS • COMPUTER SCIENCE: SD

Program Number: 21631 **Version Number:** 20123 **Effective Date:** Fall 2011

GENERAL EDUCATION

I. UNIVERSITY CORE (12 credits)	RC	CR	GR
A. Oral Communication: SPE 010 or above			
COURSE:	3		
B. Written Communication: ENG 023, 024, or 025			
COURSE:	3		
C. Mathematics: MAT 017 or above			
COURSE: MAT 140 or 301	3	**3**	
D. Wellness: HEA 110 + activity; or 3-credit Wellness course			
COURSE:	3		

II. UNIVERSITY DISTRIBUTION (15 credits)	RC	CR	GR	CAC
A. Natural Sciences: Any lab or non-lab course with prefix AST, BIO, CHM, ENV, GEL, MAR, or PHY; or certain GEG courses (see note at right)				
COURSE:	3			
B. Social Sciences: Any course with prefix ANT, CRJ, ECO, HIS, INT, MCS, PSY, POL, SOC, or SWK; or certain GEG courses (see note at right)				
COURSE:	3			
C. Humanities: Any course with prefix ENG, HUM, PAG, PHI, WRI, WST, or Modern Language				
COURSE: WRI 205	3	**3**		
D. Arts: Any course with prefix ARC, ARH, ART, CDE, CDH, CFT, DAN, FAR, FAS, MUP, MUS, or THE				
COURSE:	3			
E. Free Elective: Any course carrying university credit				
COURSE:	3			

III. COMPETENCIES ACROSS THE CURRICULUM	RC	CR	GR	CAC
A. Writing Intensive (**WI**) (9 credits)				
COURSE:	3			**WI**
COURSE:	3			**WI**
COURSE:	3			**WI**
B. Quantitative Literacy (**QL**) (3 credits) OR Computer-Intensive (**CP**) (3 credits)				
COURSE:	3			
C. Visual Literacy (**VL**) (3 credits) OR Communication-Intensive (**CM**) (3 credits)				
COURSE:	3			
D. Cultural Diversity (**CD**) (3 credits)				
COURSE:	3			**CD**
E. Critical Thinking (**CT**) (3 credits)				
COURSE:	3			**CT**

A Competency Across the Curriculum (CAC) course is not a separate course, but rather an overlay that is "double counted" as fulfilling both the CAC requirement and another requirement in either General Education (except for the University Core), the major, or the minor.

RC = Minimum required number of credits
CR = Credits earned (fill in number of credits)
GR = Grade earned (fill in letter grade)
CAC = Competency Across the Curriculum (fill in designation)

NOTE: GEG courses with a lab and 040, 322, and 323 may be used in II.A. and GEG courses 040, 204, 274, 305, 322, 323, 324, 347, 380, and 394 may NOT be used in II.B.

IV. COLLEGE DISTRIBUTION (33 credits)	RC	CR	GR	CAC
A. Natural Science, Mathematics, and Computer Science# (6 credits): Choose one course in each subcategory.				
1. Natural Science with Lab: AST, BIO, CHM, ENV, GEL, PHY, or MAR; or GEG (see note at right)				
COURSE:	3			
2. Elective: MAT, CSC, AST, BIO, CHM, ENV, GEL, PHY, or MAR; or GEG (see note at right)				
COURSE:	3			
B. Social Science (9 credits): Choose one course in each subcategory.				
1. Elective: HIS, ANT, GEG (see note at right), or POL				
COURSE:	3			
2. Elective: PSY, SOC, CRJ, or SWK				
COURSE:	3			
3. Elective: ANT, HIS, ECO, GEG (see note at right), PSY, POL, SOC, CRJ, or SWK				
COURSE:	3			

	RC	CR	GR	CAC
C. Humanities (9 credits): Choose one course in each subcategory.				
1. Elective: PAG*, ENG, WRI, or HUM				
COURSE:	3			
2. Elective: (MLS, GER, SPA, FRE, CHI, or ARA)* or PHI				
COURSE: PHI 40	3	**3**		
3. Elective: (PAG, MLS, GER, SPA, FRE, CHI, or ARA)*, ENG, WRI, HUM, or PHI				
COURSE:	3			
D. Free Electives (9 credits): Choose any university courses that count toward graduation.				
COURSE:	3			
COURSE:	3			
COURSE:	3			

NOTE: GEG courses with a lab and 040, 322, and 323 may be used in IV.A. and GEG courses 040, 204, 274, 305, 322, 323, 324, 347, 380, and 394 may NOT be used in IV.B.

Students in the College of Liberal Arts and Sciences are required to take at least one course in Biological Science (BIO) and at least one course in Physical Science (AST, CHM, ENV, GEL, PHY, MAR, GEG with lab, or GEG 040, GEG 322, or GEG 323), and at least one of which must be a lab (each course may be counted in either sections II.A or IV.A).

* Excludes courses in language instruction, conversation, composition, linguistics, syntax, stylistics, and internship courses; contact the Modern Language Studies Department for clarification.

BS/MS Computer Science: Software Development (60 SH)

Program Number: 21631 **Version Number:** 20123 **Effective Date:** Fall 2011

B. Major Program: 51 sh		
1. Required Courses: 33 sh	Gr	SH
CSC 125: Discrete Math for CS I		
CSC 225: Discrete Math for CS II		
CSC 135: Comp Sci I		
CSC 136: Comp Sci II		
CSC 235: Comp Org & Asm. Lng.		
CSC 237: Data Structures		
CSC 310: Prog. Languages		
CSC 325: Intro to CS Theory		
CSC 343: Operating Systems		
CSC 354: Software Engineering		
CSC 385: Seminar in CSC		
2. Elective Courses: 18 sh (no more than two 200-level)	Gr	SH
CSC 241: Adv Visual Basic		
CSC 242: Web Programming		
CSC 243: Java Programming		
CSC 253: IT Systems		
CSC 311: Computer Networks		
CSC 320: Intro to Game Programming		
CSC 341: Information Security		
CSC 342: Web Technologies		
CSC 351: Digital Forensics		
CSC 352: UNIX: Sys Prog/Adm		
CSC 356: Introduction to Database Sys		
CSC 402: Data Structures II		
CSC 411: Advanced Networking		
CSC 421: Web-Based Soft Design & Dev		
CSC 425: Compiler Design I		
CSC 445: Intro to Intelligent Robotics		
CSC 447: Artificial Intelligence I		
CSC 456: Database Mgm Sys I		
CSC 464: Human Computer Interaction		
CSC 480: Special Topics		

C. Concomitant Courses: 9 sh		
1. Required Courses: 9 sh	Gr	SH
MAT 171: Calculus I		
MAT 260: Linear Algebra		
MAT (elective higher than 171, not 224)		
MAT 140 or 301: Applied Stat Methods*	X	X
WRI 205: Scientific Writing*	X	X
PHI 040: Intro to Ethics*	X	X
*accounted for in general education		
2. Internship – optional (free elective)	Gr	SH
CSC 280: Cooperative Internship I		3-6
CSC 380: Cooperative Internship II		3-6
Notes on BS in Computer Science		
Before taking any 300-level course a student must have completed 18 credit hours in CSC courses numbered 125 or above with a GPA of 2.25 in the CSC courses.		
Students minoring in Math should take MAT 301		
CSC-prefix courses below 125-level, CSC 130, CSC 280 and CSC 380 do not count toward the BS in Software Development		

Internal Transfer: 2.25 GPA needed
Graduation requirement: 2.25 GPA Overall, 2.25 GPA Major
Program Code: 21631
Version Number: 20123

Approved: _____

Date: _____

Five-Year Combined BS/MS in CSC Program

120 sh	BS in Computer Science – Five Year BS/MS Undergraduate Program
30 sh	MS in Computer Science
–12 sh	Additional 12 credits of 400-level CSC courses counted toward the 120 credits for the BS above the 51 credit limit for the undergraduate degree counted for the MS
138 sh	Total credits needed to receive both the BS and MS in Computer Science

BS in CSC is awarded after the completion of 120 undergraduate credits; B-average or higher in CSC courses required for admission to Graduate School.	Program Code: 51611
	Version Number: 20093

Undergraduate admission to the Combined BS/MS Program in Computer Science: Junior status (60 or more credits taken overall) and at least 24 computer science CSC course credits with 3.00 GPA or higher and a B or better in CSC 125, 126, 235, 237, and 310 each

MS in Computer Science Portion: 30 sh

Comprehensive Examination Required for Those Not Writing a Thesis

400-level courses: 0 – 12 sh	Gr	SH
CSC 402: Data Structures II		
CSC 411: Advanced Networking		
CSC 421: Web-Based Design and Develop		
CSC 425: Compiler Design I		
CSC 445: Intro to Intelligent Robotics		
CSC 447: Artificial Intelligence I		
CSC 456: Database I		
CSC 464: Human Computer Interaction		
CSC 480: Special Topics		

500-level courses: 18 – 30 sh	Gr	SH
CSC 510: Advanced Operating Systems		
CSC 512: Network Architecture & Protocols		
CSC 516: Design & Analysis of Algors II		
CSC 520: Advanced Object Oriented Program		
CSC 521: Advanced Web-Based Soft. Devel		
CSC 526: Compiler Design II		
CSC 541: Advanced Information Security		
CSC 548: Artificial Intelligence II		
CSC 552: Advanced UNIX Programming		
CSC 555: Applied Cryptography		
CSC 556: Database II		
CSC 580: Special Topics		
CSC 599: Thesis – 6 credits		

Notes on the Five-Year Combined BS/MS in CSC:

400-level courses included on the undergraduate transcript only count here toward the masters degree if they exceed the 51 credits needed for the undergraduate degree. Internship credits are excluded from this count.

Up to 9 credits of 500-level courses may be taken during the senior year for the BS in CSC. However none of those credits count toward the 120 credits needed for the BS in CSC. They only count toward the MS in CSC.

STUDENT: _____ STUDENT ID NUMBER: _____

KUTZTOWN UNIVERSITY

I. UNIVERSITY CORE (12 credits)	RC	CR	GR
A. Oral Communication: SPE 010 or above			
COURSE:	3		
B. Written Communication: ENG 023, 024, or 025			
COURSE:	3		
C. Mathematics: MAT 017 or above			
COURSE: MAT 140	3	**3**	
D. Wellness: HEA 110 + activity; or 3-credit Wellness course			
COURSE:	3		

II. UNIVERSITY DISTRIBUTION (15 credits)	RC	CR	GR	CAC
A. Natural Sciences: Any lab or non-lab course with prefix AST, BIO, CHM, ENV, GEL, MAR, or PHY; or certain GEG courses (see note at right)				
COURSE:	3			
B. Social Sciences: Any course with prefix ANT, CRJ, ECO, HIS, INT, MCS, PSY, POL, SOC, or SWK; or certain GEG courses (see note at right)				
COURSE:	3			
C. Humanities: Any course with prefix ENG, HUM, PAG, PHI, WRI, WST, or Modern Language				
COURSE: WRI 207	3	**3**		
D. Arts: Any course with prefix ARC, ARH, ART, CDE, CDH, CFT, DAN, FAR, FAS, MUP, MUS, or THE				
COURSE:	3			
E. Free Elective: Any course carrying university credit				
COURSE:	3			

III. COMPETENCIES ACROSS THE CURRICULUM	RC	CR	GR	CAC
A. Writing Intensive (**WI**) (9 credits)				
COURSE:	3			**WI**
COURSE:	3			**WI**
COURSE:	3			**WI**
B. Quantitative Literacy (**QL**) (3 credits) OR Computer-Intensive (**CP**) (3 credits)				
COURSE:	3			
C. Visual Literacy (**VL**) (3 credits) OR Communication-Intensive (**CM**) (3 credits)				
COURSE:	3			
D. Cultural Diversity (**CD**) (3 credits)				
COURSE:	3			**CD**
E. Critical Thinking (**CT**) (3 credits)				
COURSE:	3			**CT**

COLLEGE OF LIBERAL ARTS & SCIENCES • BS/MS • COMPUTER SCIENCE: IT

Program Number:21641 **Version Number: 20123**
Effective Date: Fall 2011

GENERAL EDUCATION

A Competency Across the Curriculum (CAC) course is not a separate course, but rather an overlay that is "double counted" as fulfilling both the CAC requirement and another requirement in either General Education (except for the University Core), the major, or the minor.

RC = Minimum required number of credits
CR = Credits earned (fill in number of credits)
GR = Grade earned (fill in letter grade)
CAC = Competency Across the Curriculum (fill in designation)

IV. COLLEGE DISTRIBUTION (33 credits)	RC	CR	GR	CAC
A. Natural Science, Mathematics, and Computer Science# (6 credits): Choose one course in each subcategory.				
1. Natural Science with Lab: AST, BIO, CHM, ENV, GEL, PHY, or MAR; or GEG (see note at right)				
COURSE:	3			
2. Elective: MAT, CSC, AST, BIO, CHM, ENV, GEL, PHY, or MAR; or GEG (see note at right)				
COURSE:	3			
B. Social Science (9 credits): Choose one course in each subcategory.				
1. Elective: HIS, ANT, GEG (see note at right), or POL				
COURSE:	3			
2. Elective: PSY, SOC, CRJ, or SWK				

NOTE: GEG courses with a lab and 040, 322, and 323 may be used in II.A. and GEG courses 040, 204, 274, 305, 322, 323, 324, 347, 380, and 394 may NOT be used in II.B.

COURSE:	3			
3. Elective: ANT, HIS, ECO, GEG (see note at right), PSY, POL, SOC, CRJ, or SWK				
COURSE:	3			

	RC	CR	GR	CAC
C. Humanities (9 credits): Choose one course in each subcategory.				
1. Elective: PAG*, ENG, WRI, or HUM				
COURSE:	3			
2. Elective: (MLS, GER, SPA, FRE, CHI, or ARA)* or PHI				
COURSE: PHI 40	3	**3**		
3. Elective: (PAG, MLS, GER, SPA, FRE, CHI, or ARA)*, ENG, WRI, HUM, or PHI				
COURSE:	3			
D. Free Electives (9 credits): Choose any university courses that count toward graduation.				
COURSE:	3			
COURSE:	3			
COURSE:	3			

NOTE: GEG courses with a lab and 040, 322, and 323 may be used in IV.A. and GEG courses 040, 204, 274, 305, 322, 323, 324, 347, 380, and 394 may NOT be used in IV.B.

Students in the College of Liberal Arts and Sciences are required to take at least one course in Biological Science (BIO) and at least one course in Physical Science (AST, CHM, ENV, GEL, PHY, MAR, GEG with lab, or GEG 040, GEG 322, or GEG 323), and at least one of which must be a lab (each course may be counted in either sections II.A or IV.A).

* Excludes courses in language instruction, conversation, composition, linguistics, syntax, stylistics, and internship courses; contact the Modern Language Studies Department for clarification.

BS/MS Computer Science: Information Technology (60 SH)

Program Number: 21641 **Version Number:** 20123 **Effective Date:** Fall 2011

B. Major Program: 57 sh		
1. Required Courses: 33 sh	Gr	SH
CSC 125: Discrete Math for CS I		
CSC 130: IT Fundamentals		
CSC 135: Comp Sci I		
CSC 136: Comp Sci II		
CSC 242: Web Programming		
CSC 253: IT Systems		
CSC 311: Computer Networks		
CSC 341: Information Security		
CSC 356: Introduction to Database Sys		
CSC 354: Software Engineering		
CSC 385: Seminar in CSC		
2. Elective Courses: 24 sh	Gr	SH
CSC 225: Discrete Math for CS II		
CSC 235: Comp. Org & Assem. Lang		
CSC 237: Data Structures		
CSC 241: Adv Visual Basic		
CSC 243: Java Programming		
CSC 310: Proc Oriented Prog Lang		
CSC 320: Intro to Game Programming		
CSC 325: Intro to CS Theory		
CSC 342: Web Technologies		
CSC 343: Operating Systems		
CSC 351: Digital Forensics		
CSC 352: UNIX: Sys Prog/Adm		
CSC 402: Data Structures II		
CSC 411: Advanced Networking		
CSC 421: Web-Based Soft Design & Dev		
CSC 425: Compiler Design I		
CSC 445: Intro to Intelligent Robotics		
CSC 447: Artificial Intelligence I		
CSC 456: Database Mgm Sys I		
CSC 464: Human Computer Interaction		
CSC 480: Special Topics		

C. Concomitant Courses: 3 sh		
1. Required Courses: 3 sh	Gr	SH
MAT 105: College Algebra or above		
MAT 140: Applied Stat Methods*	X	X
WRI 207: Writing for Workplace*	X	X
PHI 040: Intro to Ethics*	X	X
*accounted for in general education		
2. Internship – optional (free elective)	Gr	SH
CSC 280: Cooperative Internship I		3-6
CSC 380: Cooperative Internship II		3-6
Recommendations		
Consider taking a Minor in an Application Domain such as Math, Psychology, Sociology, Economics, Biology, or any Science		
Consider taking a second speech course in II E		
CSC-prefix courses below 125-level, CSC 280 and CSC 380 do not count toward the BS in Information Technology		
Before taking any 300-level course a student must have completed 18 credit hours in CSC courses numbered 125 or above with a GPA of 2.25 in the CSC courses.		

Internal Transfer: 2.25 GPA needed
Graduation requirement: 2.25 GPA Overall, 2.25 GPA Major
Program Code: 21641
Version Number: 20123

Approved:

Five-Year Combined BS/MS in CSC Program

120 sh	BS in Computer Science: IT – Five Year BS/MS Undergraduate Program
30 sh	MS in Computer Science: IT
–12 sh	Additional 12 credits of 400-level CSC courses counted toward the 120 credits for the BS above the 45 credit limit for the undergraduate degree counted for the MS
138 sh	Total credits needed to receive both the BS and MS in Computer Science : IT

BS in CSC is awarded after the completion of 120 undergraduate credits; B-average or higher in CSC courses required for admission to Graduate School.	Program Code: 51621
	Version Number: 20103
Undergraduate admission to the Combined BS/MS Program in Computer Science: Junior status (60 or more credits taken overall) and at least 24 computer science CSC course credits with 3.00 GPA or higher and a B or better in CSC 125, 130, 135, 253 each	

Degree Requirements for the Master of Science in Computer Science: Information Technology track

1. Candidates for the MS degree in Computer Science must complete a total of 30 credits.
2. Students must select either the thesis option or the comprehensive exams option. The thesis option requires the completion of 24 credits of courses and 6 credits of thesis. The comprehensive exams option requires the completion of 30 credits of courses and passing the comprehensive exams. Comprehensive exams are given the first week in December, May, and August.
3. Students must complete at least 18 credits of 500-level courses.

Masters in Computer Science: Information Technology (30 sh)

At least 18 sh must be 500-level courses.

Core Courses (18 – 24 sh)	**Elective Courses (0 – 6 sh)**
CSC 411: Advanced Networking	CSC 402: Data Structures II
CSC 421: Web-Based Software Design	CSC 415: Design & Analysis of Algors I
CSC 456: Database I	CSC 425: Compiler Design I
CSC 464: Human Computer Interaction	CSC 445: Intro to Intelligent Robotics
CSC 505: Fundaments of Computer Systems	CSC 447: Artificial Intelligence I
CSC 510: Advanced Operating Systems	CSC 480: Special Topics
CSC 512: Network Architecture and Protocols	CSC 520: Advanced Object Oriented Prog.
CSC 521: Advanced Web-Based Soft. Devel.	CSC 526: Compiler Design II
CSC 541: Advanced Information Security	CSC 548: Artificial Intelligence II
CSC 552: Advanced UNIX Programming	CSC 555: Applied Cryptography
CSC 554: Project Management	CSC 556: Database II
	CSC 580: Special Topics
Thesis (0 or 6 sh)	

CSC 599: Thesis	

Department of Computer Science
MS in Computer Science: 30 sh

Comprehensive Examination Required for Those Not Writing a Thesis

400-level courses: 0 – 12 sh	Gr	SH
CSC 402: Data Structures II		
CSC 411: Networking I		
CSC 415: Design & Analysis of Algor I		
CSC 421: Web-Based Design and Develop		
CSC 425: Compiler Design I		
CSC 447: Artificial Intelligence I		
CSC 456: Database I		
CSC 464: Human Computer Interaction		
CSC 480: Special Topics		
CSC 554: Project Management		

500-level courses: 18 – 30 sh	Gr	SH
CSC 510: Advanced Operating Systems		
CSC 512: Networking II		
CSC 516: Design & Analysis of Algor II		
CSC 520: Advanced Object Oriented Program		
CSC 521: Advanced Web-Based Soft. Devel		
CSC 526: Compiler Design II		
CSC 541: Information Security II		
CSC 548: Artificial Intelligence II		
CSC 552: Advanced UNIX Programming		
CSC 555: Applied Cryptography		
CSC 557: Database II		
CSC 580: Special Topics		
CSC 599: Thesis – 6 credits		

Program Code: 51601
Version Number: 20076

Department of Computer Science
Office: Old Main 256
Phone: 610-683-4340
Kutztown University

Degree Requirements for the Master of Science in Computer Science Information Technology track

1. Candidates for the MS degree in Computer Science must complete a total of 30 credits. A candidate must complete all degree requirements for the MS within six (6) calendar years after his or her acceptance into the program.
2. Students must select either the thesis option or the comprehensive exams option. The thesis option requires the completion of 24 credits of courses and 6 credits of thesis. The comprehensive exams option requires the completion of 30 credits of courses and passing the comprehensive exams. Comprehensive exams are given the first week in December, May, and August.
3. Students must complete at least 18 credits of 500-level courses.

Masters in Computer Science: Information Technology (30 sh)

Student can take one of the following options:

- 30 sh of courses + comprehensive exams
- 24 sh of courses + 6 sh of CSC 599: Thesis

At least 18 sh must be 500-level courses.

Core Courses (18 – 24 sh)	**Elective Courses (0 – 6 sh)**
CSC 411: Networking I	CSC 402: Data Structures II
CSC 421: Web-Based Software Design	CSC 415: Design & Analysis of Algors I
CSC 456: Database I	CSC 425: Compiler Design I
CSC 464: Human Computer Interaction	CSC 447: Artificial Intelligence I
CSC 505: Fundamentals of Computer Science	CSC 480: Special Topics
CSC 510: Advanced Operating Systems	CSC 520: Advanced Object Oriented Prog.
CSC 512: Networking II	CSC 526: Compiler Design II
CSC 521: Advanced Web-Based Soft. Devel.	CSC 548: Artificial Intelligence II
CSC 541: Advanced Information Security	CSC 555: Applied Cryptography
CSC 552: Advanced UNIX Programming	CSC 580: Special Topics
CSC 554: Project Management	CSC 557: Database II
Thesis (0 or 6 sh)	
CSC 599: Thesis	

Program Code: 51602
Version: 20123
Approval:

Master of Science in Computer Science
PSM Track

Candidates must complete a total of 33 credits.

A candidate must complete all degree requirements for an MS within six (6) calendar years after his or her acceptance. Requirements for the MS CS – PSM Track are:

- 15-21 hours of CS core courses
- 0 – 6 hours of CS elective courses
- 3 hours of CS internship
- 9 hours of Professional Skills courses

This includes the following requirements:

- A minimum of 15 hours of CSC courses must be at the 500 level
- Successful completion of **one** of the following concentrations:
  - Networking: CSC 411 and CSC 512
  - Security: CSC 541 and CSC 555
  - Management: CSC 554 and BUS 572
  - Web-based Software: CSC 421 and CSC 521
  - Database CSC 456 and CSC 556

Core CS Courses (15 – 21 hours)	**Elective CS Courses (0 – 6 hours)**
CSC 411: Advanced Networking	CSC 402: Data Structures II
CSC 421: Web-based Software Development	CSC 425: Compiler Design I
CSC 456: Database Systems Management I	CSC 445: Intelligent Robotics
CSC 464: Human Computer Interaction	CSC 447: Artificial Intelligence I
CSC 505: Fundamentals of Computer Systems	CSC 480: Special Topics
CSC 510: Advanced Operating Systems	CSC 526: Compiler Design II
CSC 512: Network Architecture & Protocols	CSC548: Artificial Intelligence II
CSC 520: Advanced Object Oriented Prog.	CSC 552: Advanced UNIX Programming
CSC 521: Advanced Web-Based Soft. Dev.	CSC 555: Applied Cryptography
CSC 540: Engineering Enterprise Object Oriented Systems	CSC 556: Database Systems Management II
CSC 541: Advanced Information Security	CSC 580: Special Topics
CSC 554: Project Management	
CS Culminating Experience (3 hours)	
CSC 590: Internship (3 hours)	

Professional Skills Courses

Professional skills courses provide breadth in graduate studies by permitting study in areas that enhance and broaden the student's knowledge and skills. The Computer Science PSM Track provides the opportunity to incorporate study in management and administration related disciplines into a computer science Master's degree

Courses may be selected from Kutztown or other PASSHE universities to fulfill this category under the following conditions:

1. Courses selected must develop skills in related professional areas such as management, communication, planning, and decision making. They need not all be from a single discipline.
2. The courses selected represent a cogent concentration and are chosen and approved in consultation with the graduate coordinator and department chair.

Professional Skill Courses (9 hours)
Business
ACC 520: Management Accounting Control Systems
BUS 572: Managing Enterprise Information Technology
ECO 561: The Global Economy

Political Science – Administration
POL 515: Principles of Problems in Public Administration
POL 516: Administrative and Organizational Theory
POL 517: Public Personnel Administration and Strategic Human Resource Management
POL 518: Budgeting and Decision Making
POL 571: Nonprofit Organizational Management

Elective(s)
Elective:
Elective:
Elective:

Computer Science Minor in Software Development

Minor Program: 21 sh		
1. Required Courses – 15 sh	Gr	SH
CSC 125: Discrete Math for Computing I		
CSC 225: Discrete Math for Computing II		
CSC 135: Computer Science I		
CSC 136: Computer Science II		
CSC 237: Data Structures		
2. Elective Course – 3 sh	Gr	SH
Any 200-level or higher CSC* course		
3. Elective Course – 3 sh		
Any 300-level or higher CSC ** course		

Total Semester Hours	
Required Courses	**15**
Elective Course	**6**
TOTAL	**21**

Software Development Minor
Program # 00074
Version # 20123
Effective Date:Fall 2011, revised fall 2011

Approved _____ Date _____

*If you are thinking of going on for an MS in Computer Science (software development) you should take CSC 235.

** If you are thinking of going on for an MS in Computer Science (software development) you should take CSC 310.

Computer Science Minor in Information Technology

Minor Program: 21 sh		
1. Required Courses – 15 sh	Gr	SH
CSC 125: Discrete Math for Computing I		
CSC 130: IT Fundamentals		
CSC 135: Computer Science I		
CSC 136: Computer Science II		
CSC 253: IT Systems		
2. Elective Course – 3 sh	Gr	SH
Any 200-level or higher CSC course		
3. Elective Course – 3 sh		
Any 300-level or higher CSC** course		

Total Semester Hours	
Required Courses	**15**
Elective Course	**6**
TOTAL	**21**

Information Technology Minor
Program # 00035
Version # 20123
Effective Date: Fall 2011, revised fall 2011
Approved: _____ Date: _____

**If you are considering an MS in Information Technology, you should take one of the following: CSC 311, CSC 341 or CSC 352.

Appendix B Recommended Course Sequences

Suggested Schedule for Software Development Students

Freshman Year – Fall	Freshman Year - Spring
CSC 135 MAT 181 ENG 023	CSC 136 CSC 125 CSC 242 or 241 PHI 040
Sophomore Year – Fall	Sophomore Year – Spring
CSC 237 CSC 225 CSC 235 PHI 140	CSC Elective CSC 325 MAT 140 (301 for math minors) Science Elective
Junior Year – Fall	Junior Year – Spring
CSC 343 CSC Elective WRI 205 2nd Science Elective	CSC 310 CSC Elective MAT 260
Senior Year – Fall	Senior Year - Spring
CSC 354 CSC Elective MAT Elective	CSC 385 CSC Elective

Suggested Schedule for Information Technology Students

Freshman Year – Fall	Freshman Year - Spring
CSC 135 MAT 105 CSC 130 ENG 023	CSC 136 CSC 125 CSC 242
Sophomore Year – Fall	Sophomore Year – Spring
CSC 253 CSC Elective MAT 140	CSC 356 CSC Elective
Junior Year – Fall	Junior Year – Spring
CSC 311 CSC 341 CSC Elective WRI 207	CSC Elective CSC Elective MAT 140
Senior Year – Fall	Senior Year - Spring
CSC 354 CSC Elective CSC Elective	CSC 385 CSC Elective

Appendix C

Tentative Course Offerings

This appendix contains the tentative three-year rotation for courses. It is important to note that the course offerings are subject to change as student enrollment, student's needs and faculty resources are considered.

	Fall11	Spring12	Fall12	Spring13	Fall13	Spring14	
Course	**Fall 2**	**Spring 2**	**Fall 3**	**Spring 3**	**Fall 4**	**Spring 4**	
010 - Computer Applications	x	x	x	x	x	x	
020 - Computer Graphics	x	x	x	x	x	x	
111 - computer forensics	x	x	x	x	x	x	
112 - VB	x	x	x	x	x	x	
122 - Web Design	x	x	x	x	x	x	
125 - Discrete	x	x	x	x	x	x	
130 IT Fundmentals	x	x	x	x	x	x	
135 - Prog I	x	x	x	x	x	x	
136 - Prog II	x	x	x	x	x	x	
225 - Discrete II	x	x	x	x	x	x	
235 - Organization	x	x	x	x	x	x	
237 - DataStuctures	x	x	x	x	x	x	
241 - Advanced VB	x	x	x	x	x	x	
242 - Web Programming	x	x	x	x	x	x	
243 - Java		x		x		x	
253 - Intro To IT Systems	x	x	x	x	x	x	
310 - Programming Lang		x		x		x	
311 - Intro to Networks	x		x		x		
320 - Game Design	x		x		x		
325 - Intro to CS Theory		x		x		x	
341 - Security	x		x		x		
342 - Web Technologies		x		x		x	
343 - Operating Systems	x	x	x		x	x	
351 - Digital Forensics		x		x		x	
352 - UNIX	x	x		x	x	x	
354 - Software Engineering	x	x	x	x	x	x	
356 - Database I		x		x		x	
385 - Senior Sem	x	x	x	x	x	x	
402 - Data Structures II	x		x		x		
411 - Advanced Networking		x		x		x	
415 - Algorithms		x		x		x	
421 - Web Based Software Design		x		x		x	
425 - Compiler	x				x		
445 - Intelligent Robotics			x				
447 - AI		x		x		x	
456 - Data Base II	x		x		x		
464 - Human Computer Design							summers
480 - Selected Topics		When appropriate					
510 - Operating Sys II		x				x	
512 - Networking II	x				x		
516 - Algorithms II			x				
520 - Advanced OO			x				
521 - Web Applications	x				x		
526 - Compiler II		x				x	
541 - Advanced Security				x			
548 - AI II	x				x		
552 - Advanced UNIX			x				
554 - Project Management							
555 - Applied Cryptography		x				x	
557 - Database II				x			
580 - Special Topics				x			